Eur. Phys. J. C (2020) 80:230
https://doi.org/10.1140/epjc/s10052-020-7786-0

THE EUROPEAN PHYSICAL JOURNAL C

Special Article - New Tools and Techniques

HPGe detector field calculation methods demonstrated with an educational program, GeFiCa

Jianchen Li, Jing Liu[a], **Kyler Kooi**

Department of Physics, University of South Dakota, 414 East Clark Street, Vermillion, SD 57069, USA

Received: 10 January 2020 / Accepted: 26 February 2020 / Published online: 11 March 2020

Abstract A review of tools and methods to calculate electrostatic potentials and fields inside high-purity germanium detectors in various configurations is given. The methods are illustrated concretely with a new educational program named GeFiCa - Germanium detector Field Calculator. Demonstrated in GeFiCa are generic numerical calculations based on the successive over-relaxation method as well as analytic ones whenever simplification is possible due to highly symmetric detector geometries. GeFiCa is written in C++ and provided as an extension to the CERN ROOT libraries widely used in the particle physics community. Calculation codes for individual detectors, provided as ROOT macros and python scripts, are distributed along with the GeFiCa core library, serving as both examples showing the usage of GeFiCa and starting points for customized calculations. They can be run without compilation in a ROOT interactive session or directly from a Linux shell. The numerical results are saved in a ROOT tree, making full use of the I/O optimization and plotting functionalities in ROOT. The speed and precision of the calculation are comparable to other commonly used packages, which qualifies GeFiCa as a scientific research tool. However, the main focus of GeFiCa is to clearly explain and demonstrate the analytic and numeric methods to solve Poisson's equation, practical coding considerations and visualization methods, with intensive documentation and example macros. It serves as a one-stop resource for people who want to understand the operating mechanism of such a package under the hood.

1 Introduction

The calculation of electrostatic potentials and fields in a high-purity germanium (HPGe) detector is the initial step in a full pulse-shape simulation [1–5] process. It is also used to guide the design of novel detector geometries to avoid unreasonably high depletion voltages or hidden undepleted regions, which may occur when the size of a detector is enlarged [6]. The design of read-out electronics can benefit from it as well since the capacitance of a detector, a determination factor of the electronics noise, can be calculated from the energy stored in the electric field in the detector. It is widely used in HPGe detector based neutrinoless double beta ($0\nu\beta\beta$) decay experiments, such as GERDA [7] and MJD [8], dark matter experiments, such as CoGeNT [9], Texono [10] and CDEX [11], and gamma-ray tracking detectors to study structures of atomic nuclei, such as AGATA [12] and GRETA [13], etc.

A complete list is impossible, but commonly used field calculation packages include fieldgen (an essential part of siggen [1,14]) used in GRETINA [15] (an early phase of GRETA) and MJD, ADL [2] and SIMION [2–4,16] used in AGATA and GERDA, Maxwell [17] used in most experiments, MaGe [18] used in GERDA and MJD, and FEniCS [19], a popular open-source computing platform for solving partial differential equations. A new package called *SolidStateDetectors.jl* [20], SSD in short hereafter, is under rapid development at the Max-Planck-Institut für Physik for LEGEND [21], a new $0\nu\beta\beta$ experiment as a combined effort of GERDA and MJD.

SIMION is a commercial software package primarily used to simulate the transportation of charged particles in static or low-frequency RF fields. According to its documentation [16], it uses the finite-element method to calculate 2D and 3D fields with up to almost 20 billion grid points, given enough RAM. Its power in static field calculation is overkill for common HPGe detector configurations while lacking some important features that are required for HPGe detector applications, such as the calculation of depletion voltage, region and detector capacitance, etc. This is understandable given that the main application of SIMION is not HPGe detector field calculation.

[a] e-mail: Jing.Liu@usd.edu (corresponding author)

ADL stands for AGATA Detector Library [2]. It is used by the AGATA collaboration to simulate and analyze pulse shapes of electronic signals from segmented HPGe detector arrays in order to determine interaction positions of γ-rays originated from the nuclear target under study. It can be used to calculate electric fields in common detector configurations. However, AGATA detectors take irregular shapes to be tightly packaged together, their fields are calculated using SIMION.

ANSYS Maxwell [17] is a more popular electromagnetic field simulation software compared to SIMION. It is for the design and analysis of electric motors, actuators, sensors, transformers and other electromagnetic and electromechanical devices. It uses automatic adaptive meshing techniques to achieve user-specified accuracy without detailed instruction from a user. As its main application is not HPGe detector field calculation, it has the same advantages and disadvantages as SIMION, but is more expensive than SIMION.

A common pitfall of all general-purpose commercial software is that one has to pay for extra features that are not needed in the HPGe field calculation, while still missing out some basic features that are needed.

FEniCS [19], on the other hand, is a free-to-use, open-source program developed by a global community of scientists and software developers, and is just as sophisticated as SIMION and Maxwell. Using efficient finite-element codes, its main purpose is to solve partial differential equations, including Poisson's equation, which is needed in HPGe field calculations. As versatile as it is, FEniCS demands effort to adapt it to a specific application, such as calculating fields in HPGe detectors. From this point of view, FEniCS has the same drawback as commercial packages, that is, it is overkill for HPGe field calculation, but lacks basic features that are specific for HPGe application. Nonetheless, there is ongoing effort within the MJD collaboration to adapt it for HPGe detectors.

On the contrary, MaGe [18] and siggen [1,14] are dedicated software for HPGe signal formation simulation. They are not as versatile and sophisticated as the previously mentioned packages, but are sufficient for the HPGe application. Initially, MaGe was jointly developed by the Majorana [8] and GERDA collaborations mainly as a GEANT4 [22–24] based Monte Carlo simulation package. It was extended later on to include a full pulse-shape simulation chain using GEANT4 simulation results as input [5,25,26]. It can be used for the simulation of both segmented [27] and point-contact [28] detectors. The major drawback of MaGe is that it is only available for the GERDA or Majorana collaborators.

Siggen [1,14] is mainly developed by David Radford for MJD pulse-shape simulation. It is open-source and free to use. A stand-alone portion of siggen, called fieldgen, is dedicated to the calculation of fields and potentials of point-contact detectors in two dimensional cylindrical coordinates. It cannot be used for segmented detectors. The program is written in c, but the configuration file is in plain ASCII with straightforward syntax for a user to easily specify detailed dimensions of a detector, such as the size of small electronic contact, or the width of a groove to reduce surface leakage current. Fieldgen can also be used to calculate the capacitance of a detector, the full depletion voltage, and the depletion region in case that a detector is not fully depleted. Those functions are not available in the packages mentioned previously. Fieldgen utilizes the successive over-relaxation method (SOR) to first calculate the potential in a coarse grid with a typical distance of 1 mm between two grid points. The result of this coarse calculation is then used as the input of a more precise calculation in a finer grid with a typical distance of 0.1 mm between two grid points. Using this simple approach in place of automatic adaptive meshing techniques used in some of the other packages makes fieldgen both fast and accurate enough for its dedicated application.

SSD [20] is mainly developed by the GeDet group at the Max-Planck-Institut für Physik for LEGEND [21]. It is capable of not only the calculation of electric fields but also the simulation of electronic signals. In its field calculation part, it contains functions to deal with common detector configurations, such as point-contact and segmented ones. It features adaptive grid sizes, which improves both the calculation speed and accuracy. It is written in *Julia* [29], a relatively new, high-level, general-purpose programming language designed to address the needs of high-performance numerical analysis. It is possible to enable multi-threading in SSD, which takes the full advantage of modern computer hardware. Compared to MaGe and fieldgen, SSD has an attractive feature to calculate the field outside of a detector taking into account the influence of the detector holding structure nearby.

Overall, fieldgen seems to be the maturest at this moment for users interested in HPGe field calculation as long as their detector geometry is similar to that of point-contact ones.[1] However, the lack of detailed documentation makes it hard for a developer to modify the code of fieldgen for other geometries or to add new features.

This is one of the reasons why many research groups write their own code for HPGe detector field calculation instead of using the mentioned major players. An obvious advantage of home brewed code is that it is well understood and easy to tune if needed. The second advantage is that writing their own code instead of using existing ones deepens the understanding of junior researchers on HPGe detector working principles and numerical calculation techniques. Drawbacks of this approach include the limited functionality, the lack of verification and the waste of time in reinventing the wheel.

[1] It is not as limiting as it sounds, because the bore hole of a common coaxial detector can be regarded as a very large point-contact, and a planar detector can be regarded as having a large flat point-contact. The main limitation is on segmented or stripped contacts.

GeFiCa is aimed at clear explanation and demonstration of the analytic and numeric methods to solve Poisson's equation, practical coding considerations and visualization methods. It does so by providing intensive documentation and example macros, and serves as a one-stop resource for people who want to understand the operating mechanism of such a package under the hood. None of the tools mentioned above fits all applications. Home brewed codes built on top of some existing tools may be the best choice for education and specific applications, as long as the drawbacks mentioned previously can be effectively overcome through the demonstration provided in GeFiCa.

2 Space charges

HPGe crystals come in two types. As shown in Fig. 1, if the trace impurity atoms in a crystal provide free-moving electrons (phosphorus, for example), the crystal is of n-type, and if the atoms provide free-moving holes (boron, for example), the crystal is of p-type. In both literature and popular science articles, these free-moving charge carriers are often preceded with adjectives like "extra" or "excess", which may lead to a false impression that an n-type crystal has "extra" electrons donated by donor impurity atoms and is hence negatively charged, or that a p-type crystal has "extra" holes (vacancies in covalent bonds) due to acceptor impurity atoms and is positively charged. These free-moving charges can be regarded as "extra" since they are not used in forming covalent bonds between atoms, which is the fundamental reason why they are free. But they are not "extra" charges that break the balance of the numbers of protons and electrons in a crystal. Actually, no matter which type it is, a crystal is electrically neutral as a whole because the number of protons are the same as the number of electrons in both impurity and Ge atoms.

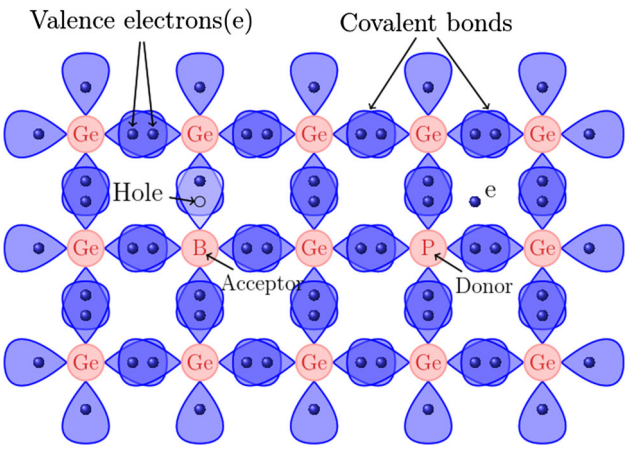

Fig. 1 Conceptual sketch of covalent bonds between Ge and impurity atoms (P and B, as examples)

As trivial as it sounds, this fact is worthy of emphasizing, especially for one to understand the sign of space charges to be mentioned in the following paragraph.

When the bias voltage applied to a crystal is high enough, all free-moving charge carriers can be swept out of the bulk of the crystal. The crystal is said to be depleted of free charge carriers. In an n-type crystal, it is the free-moving electrons that are swept out. Consequently, the trace impurity atoms are positively ionized. Since the ions are fixed in their locations in the crystal, they cannot be swept out by the external electric field, and are hence called "space charges". In a p-type crystal, however, the space charges are negative, since it is the free-moving holes that are swept out. It is quite counter intuitive for one to realize the fact that a depleted n-type crystal is actually positively charged and a p-type negatively charged. Space charges create an electric field in addition to the one that is created by the bias voltage. The total electric field inside a depleted crystal is the sum of these two.

The space charge density distribution, normally denoted as ρ, can be quite complicated due to the nature of HPGe single crystal growth process [30,31]. It is normally characterized in the following way. First, a few wafers are cut from various axial positions in a HPGe single-crystal boule pulled using the Czochralski method, typically, one from the shoulder and one from the tail of the boule. Second, small samples are cut from individual wafers along their radius. Net impurity concentrations of these samples, $N_A - N_D$, are then measured using Hall-effect, where N_A is the acceptor concentration, N_D donor concentration. Since there is a relationship between ρ and $N_A - N_D$ as explained in the previous paragraph:

$$\rho = -(N_A - N_D)e, \tag{1}$$

where $e = 1.6 \times 10^{-19}$ C is the elementary charge, both the vertical (axial) and radial distributions of ρ can be investigated this way (Fig. 2).

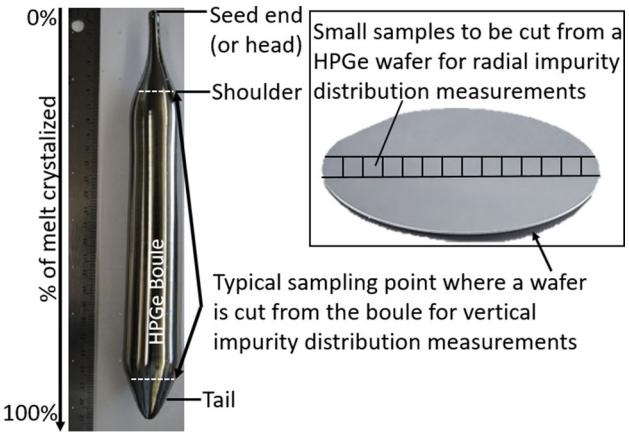

Fig. 2 A HPGe single-crystal boule pulled using the Czochralski method, and a HPGe wafer cut from the boule for impurity measurements

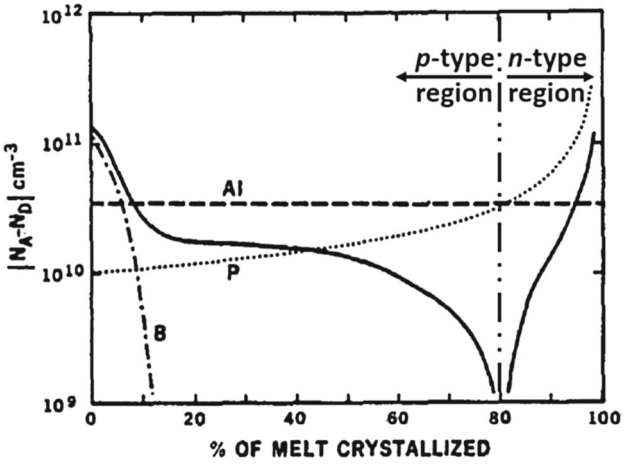

Fig. 3 A typical vertical net impurity concentration profile of a HPGe single-crystal boule, taken from [30]

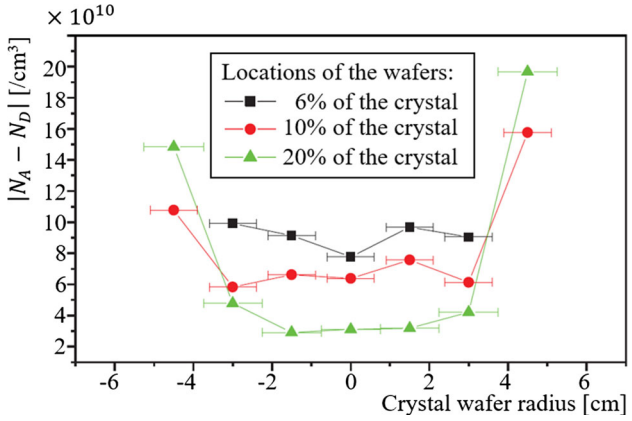

Fig. 4 A typical radial net impurity concentration profile of a HPGe single-crystal boule, taken from [31]

A typical vertical net impurity concentration profile of a HPGe single-crystal boule is shown in Fig. 3 taken from [30] with a vertical double-dotted dashed line added to clearly indicate the p-type and n-type regions. The dashed line indicates the contribution from a typical p-type impurity element, Al. The dotted dashed curve indicates the contribution from another typical p-type impurity element, B. The dotted curve shows the contribution from a typical n-type impurity element, P. The solid curve broken around 80% of the boule is the overall net impurity concentration. The crystal is of p-type from 0 to 80% of its length, and changes to n-type after that. The curve is approximately flat from 20 to 40%, which is a typical portion of the boule to be harvested for detector fabrication.

A typical radial net impurity concentration profile of a HPGe single crystal is shown in Fig. 4 taken from [31]. It is basically flat from 0 to a certain radius, but increases dramatically close to the skin of the crystal. Sometimes, a crystal may even change its type from its center to its outer radius, as mentioned in [30]. The skin of a boule may be removed so that the central part used for detector fabrication has a relatively constant impurity distribution.

Given those experimental evidences, the space charge density in general has to be expressed as a function of location, i.e., $\rho(\mathbf{x})$, where $\mathbf{x}$ is a vector indicating the location of interest. Since the measurement of impurity is destructive for the raw material, the real impurity distribution in a crystal used for detector fabrication is usually unknown. Normally, only the average impurities close to the top and bottom of the cut portion of the crystal are known. The impurity distribution in between is regarded as a constant or approximated by a first-order polynomial determined by the average top and bottom impurities. If the right portion of a crystal (20–40% of the black line in Fig. 3, for example) is harvested for detector fabrication, this is normally an acceptable approximation. However, one has to keep in mind that our knowledge of the real impurity distribution is incomplete, and our approximation may have sizable uncertainties.

3 Poisson's equation

The existence of space charges complicates the calculation of the electrostatic potential in a HPGe detector. Without space charges, the potential can be calculated by solving Laplace's equation,

$$\nabla^2 V(\mathbf{x}) = 0, \tag{2}$$

where $V(\mathbf{x})$ is the potential to be determined. With space charges, however, the potential must be calculated by solving Poisson's equation, which takes into account the space charge distribution in the bulk of a detector:

$$\nabla^2 V(\mathbf{x}) = -\frac{\rho(\mathbf{x})}{\epsilon}, \tag{3}$$

where $\epsilon = \epsilon_0 \epsilon_r$ with $\epsilon_0 \approx 8.854 \times 10^{-12}$ F/m being the permittivity in vacuum, and $\epsilon_r \approx 16.0$ being the relative permittivity (or dielectric constant) of Ge.

Both differential equations have an infinite amount of solutions characterized by a few undetermined constants. These constants can be fixed by boundary conditions, which refers to the voltage values on detector electrodes. The relationship between these two equations can be understood better when we consider two different boundary condition setups: first, potentials of electrodes of a detector are set based on the bias voltage applied to the detector, second, they are all set to zero. If Laplace's equation is solved with the first setup, its solution is a potential field caused by the bias only. If Poisson's equation is solved with the second setup, its solution is simply the potential caused by the space charges only. The potential in a detector is a linear combination of these two solutions. We can also solve Poisson's equation with the

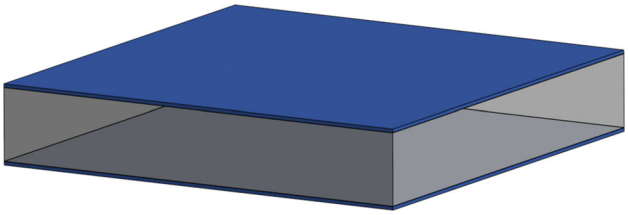

Fig. 5 3D model of a planar detector with electrodes indicated with blue

first set of boundary conditions, which directly results in the combined potential

In addition to the potential, we are also interested in the electric field distribution in a detector. The electric field vector **E** can be then determined with the equation

$$\mathbf{E} = -\nabla V. \tag{4}$$

These equations are rather abstract. A concrete expression can be obtained in a specific coordinate system. For example, in Cartesian coordinates, Poisson's equation reads,

$$\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2} = -\frac{\rho(x, y, z)}{\epsilon}, \tag{5}$$

where, x, y, z are the three Cartesian coordinates. The expression of Poisson's equation in spherical and cylindrical coordinates are listed in "Appendix A" (Fig. 5).

4 Analytic solutions

4.1 Planar detectors

As mentioned in Sect. 2, in general, the space charge density ρ is a function of **x**, and there is no analytic solution for three dimensional Poisson's equation with a complicated $\rho(\mathbf{x})$. However, in certain highly symmetric detector configurations, Poisson's equation can be significantly simplified and its analytic solution can be obtained. For example, at the center of a large but thin planar HPGe detector, the electric potential can be regarded as only varying along the thickness of the detector, x, and ρ can be regarded as a constant. Eq. (5) can then be simplified to

$$\frac{\mathrm{d}^2 V}{\mathrm{d}x^2} = -\frac{\rho}{\epsilon}. \tag{6}$$

Its analytic solution reads,

$$V = -\frac{\rho}{2\epsilon}x^2 + C_2 x + C_1, \tag{7}$$

where C_1 and C_2 are constants which can be determined using two boundary conditions, i.e., the voltages of two planar detector electrodes. The electric field reads,

$$E = -\frac{\mathrm{d}V}{\mathrm{d}x} = \frac{\rho}{\epsilon}x - C_2, \tag{8}$$

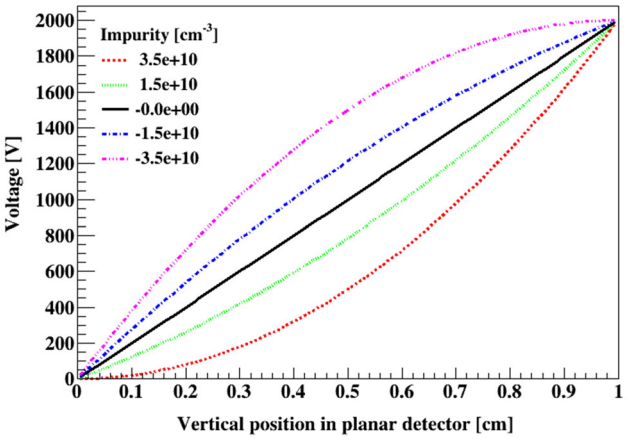

Fig. 6 Voltage distributions in an ideal planar detector

Figure 6 shows the voltage as a function of the vertical position in an ideal planar detector, assuming a thickness of 1 cm and a voltage of 2000 V applied to its top electrode. The net impurity concentration corresponding to each curve in the figure is listed in the legend. When $\rho = 0$, Eq. (7) becomes $V = C_2 x + C_1$, which is simply a straight line between [0, 0] and [1 cm, 2000 V]. The higher an impurity concentration, the more a curve is bent up or down depending on the type of the impurity. Since the slope of curves in Fig. 6 is proportional to the magnitude of the electric field, as shown in Eq. (8), the bending of the curves shows how space charges modify the overall electric field in a detector. This is demonstrated explicitly in Fig. 7, where the y-axis changes to electrical field in the unit of V/cm. A small change of the impurity concentration may result in large deviation of the overall field from the constant external field. When the impurity concentration is high enough, the electric field close to the electrodes of the detector can be as low as zero. Such low field regions are where severe charge trapping may happen, which deteriorates the energy resolution of the detector, hence are not desirable. Obvious solutions of such a problem include applying a voltage significantly higher than the depletion voltage, reducing the thickness of the detector, or growing purer crystals. The first solution is dangerous, the second is undesirable and the last is difficult. A less obvious alternative is to switch to a different geometric configuration of the detector.

4.2 Coaxial detectors

How the geometry of a detector can help solve this problem can be clearly demonstrated using the analytic solution of Poisson's equation in cylindrical coordinates (Fig. 8). The electric potential far away from its two end surfaces of a true-coaxial HPGe detector can be regarded as varying only with r. If one further assumes that the space charge density

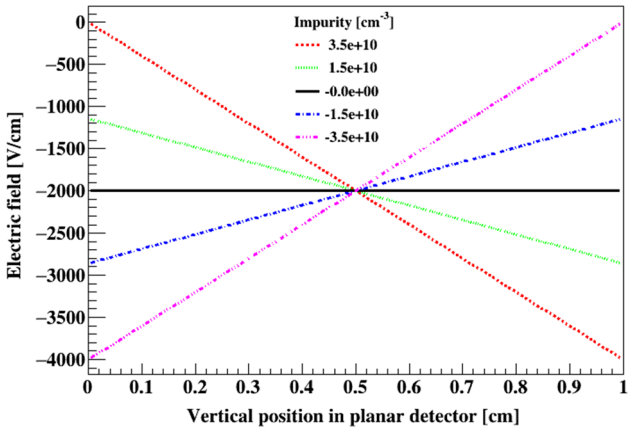

Fig. 7 Electric field distributions in an ideal planar detector

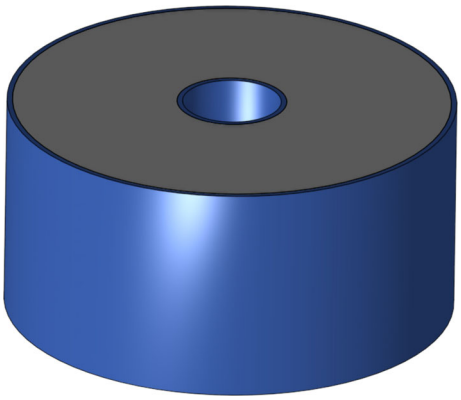

Fig. 8 3D model of true coaxial detector with electrodes indicated with blue

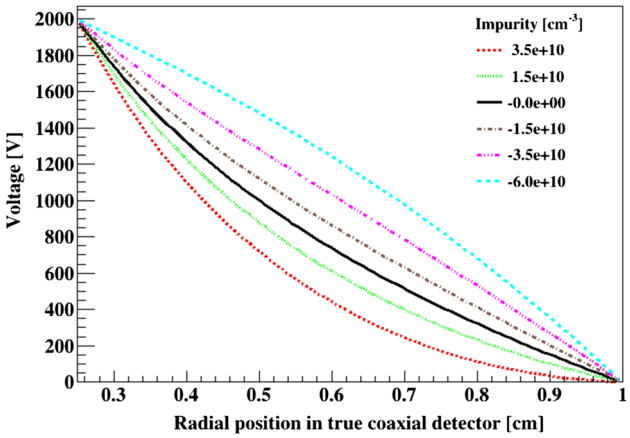

Fig. 9 Voltage distributions in a true coaxial detector

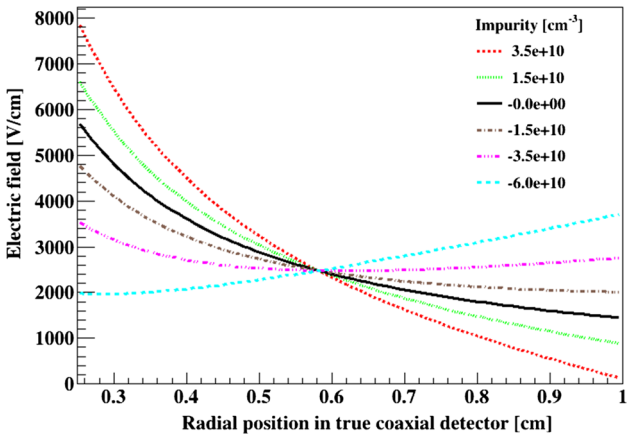

Fig. 10 Electric field distributions in a true coaxial detector

ρ is a constant, Poisson's equation in cylindrical coordinates can be simplified to

$$\frac{1}{r}\frac{\mathrm{d}}{\mathrm{d}r}\left(r\frac{\mathrm{d}V}{\mathrm{d}r}\right) = -\frac{\rho}{\epsilon}. \tag{9}$$

Its analytic solution reads,

$$V = -\frac{\rho r^2}{4\epsilon} + C_1 log(r) + C_2, \tag{10}$$

where C_1 and C_2 are constants, which can be determined using boundary conditions, that is, the locations and voltages of the two electrodes of a detector. The electric field is then

$$\mathbf{E} = -\nabla V = \frac{\rho r}{2\epsilon} - \frac{C_1}{r}. \tag{11}$$

Figure 9 shows the voltage as a function of the radial position in a true-coaxial detector with an inner radius of 0.25 cm, an outer radius of 1 cm and a voltage of 2000 V applied to its inner electrode. The net impurity concentration corresponding to each curve in the figure is listed in the legend. Compared to Fig. 6 for a planar detector, the curve corresponding to the zero impurity is not a straight line anymore. Instead, it bends downward, reflecting the fact that the electrical field close to the inner radius is stronger, which can be seen in Fig. 10 as well. Now, p-type impurities bend the curves further down, while n-type impurities bend them upwards, effectively flatten the electrical field distributions along the radius, as shown explicitly in Fig. 10. Given the right impurity concentration, the electrical field in a true-coaxial detector can be optimized to avoid charge trapping. To demonstrate this point more clearly, a curve corresponding to a high impurity concentration of -6×10^{10}/cm^3 is added to Figs. 9 and 10, which is not in Figs. 6 and 7. The electric field distribution corresponding to this concentration is in between $\sim$ 2000 V/cm and 4000 V/cm, well above zero in the entire volume of the detector. One has to avoid falling into a false impression that coaxial detectors prefer n-type crystals to p-type ones. In reality, this preference can be easily flipped by flipping the bias polarity. It is better to say that the type of the crystal prefers a certain bias polarity. As a conclusion, coaxial detectors are much more tolerant of high impurity concentrations than planar ones (Fig. 11).

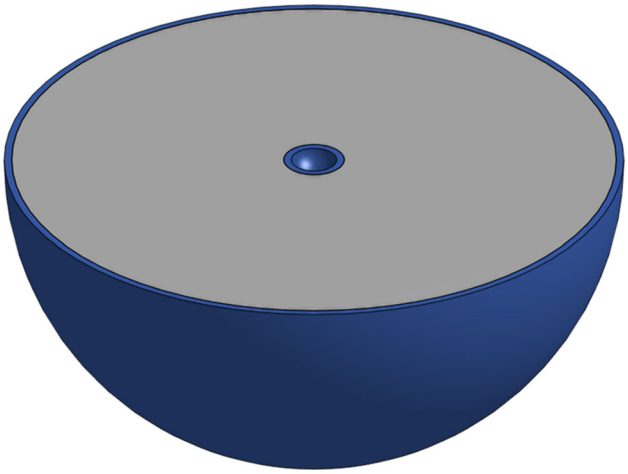

Fig. 11 3D model of a hemispherical detector with electrodes indicated with blue

4.3 Hemispherical detectors

In spherical coordinates with polar and azimuthal symmetries, the θ and ϕ terms can be dropped and the Poisson's equation can be simplify to

$$\frac{1}{r^2}\frac{\mathrm{d}}{\mathrm{d}r}\left(r^2\frac{\mathrm{d}V}{\mathrm{d}r}\right) = -\frac{\rho}{\epsilon}. \tag{12}$$

Assuming constant ρ, its analytic solution reads,

$$V = -\frac{\rho r^2}{6\epsilon} + C_1\frac{1}{r} + C_2, \tag{13}$$

where C_1 and C_2 are constants that can be determined by boundary conditions. The electrical field

$$\mathbf{E} = -\nabla V = \frac{\rho r}{3\epsilon} + C_1\frac{1}{r^2}. \tag{14}$$

A detector with such a configuration is more impurity tolerant than a coaxial one. However, it is not easy to bias the inner radius of a full sphere. The electrical field of a hemispherical detector can be approximated with the same solution, and it is possible to apply voltage to its inner radius. However, it is a significant challenge to machine a cylindrical single-crystal boule into such a shape. For this reason, no hemispheric HPGe detectors have been made so far, and most HPGe detectors take the cylindrical shape for convenience.

If the inner radius of an imaginary hemispheric detector is small enough, say, about 1 mm, it can help illustrate some important properties of a point-contact detector, which can be imagined as a traditional coaxial detector with its central contact shrunk to a point. For example, certain amount of impurity is necessary to shape the electrical field in the detector so that it is not too strong close to the point-contact and not too weak far away from it. This is why the first point-contact detector is called a "shaped-field" one [28].

4.4 Depletion voltage

Given fixed dimensions and impurity concentration of a crystal, we'd like to find out the voltage at which the crystal can be fully depleted, or the depletion voltage, V_d. The method to solve this problem can be demonstrated using the analytic solution, Eq. (7), of the one-dimensional Poisson's equation in Cartesian coordinates. The strategy can be applied to multi-dimensional configurations with minor modifications.

To keep our discussion as concrete as possible, let us assume an ideal planar detector with a thickness of $d = 1$ cm and a homogeneous impurity concentration of $4 \times 10^{10}/\mathrm{cm}^3$ (p-type) in its entire volume. Let's further assume that its bottom electrode is at $x = 0$ and grounded, i.e.

$$V(x = 0) = 0 \tag{15}$$

Applying this boundary condition to Eq. (7), we have $C_1 = 0$. If no bias is applied at the top electrode, that is, $V(x = d) = 0$, we can further get $C_2 = \rho d/(2\epsilon)$, where $\rho = -4 \times 10^{10}/\mathrm{cm}^3 e$ is the space charge density. Eq. (7) can then be rearranged as,

$$V(x) = -\frac{\rho}{2\epsilon}\left(x - \frac{d}{2}\right)^2 + \frac{\rho d^2}{8\epsilon}, \tag{16}$$

which is the green parabola shown in Fig. 12. However, this is not physically correct, since the whole crystal should be at $V = 0$ without any bias. The problem comes from our taken-as-granted assumption that $\rho = -4 \times 10^{10}/\mathrm{cm}^3 e$, which is only true in depleted region. In undepleted region, there should not be any space charge, that is, $\rho = 0$, which indeed guarantees $V(x) = 0$ in Eq. (16).

Instead of regarding it as a mistake, there is a better way to interpret Eq. (16), that is, it is the contribution to the voltage from the "space charge alone", when the detector is fully depleted. Its value hence is not dependent on the bias voltage after fully depletion. The overall voltage should be the sum of this contribution and the voltage due to the external bias.

The external bias voltage distribution without any contribution from space charges is simply $V(x) = C_2 x$ (Eq. 7 with $\rho = 0, C_1 = 0$), that is, a straight line, as the one labelled "bias alone" in Fig. 12.

At a bias voltage of -1000 V, the sum of "space charge alone" and "bias alone" contributions gives the red curve in Fig. 12 that is below the -1000 V line in a wide region. This region can be regarded as a potential well where positive free charge carriers are trapped. In another word, the -1000 V bias is not enough to sweep all free charges out of the crystal. The curve is hence labelled "undepleted".

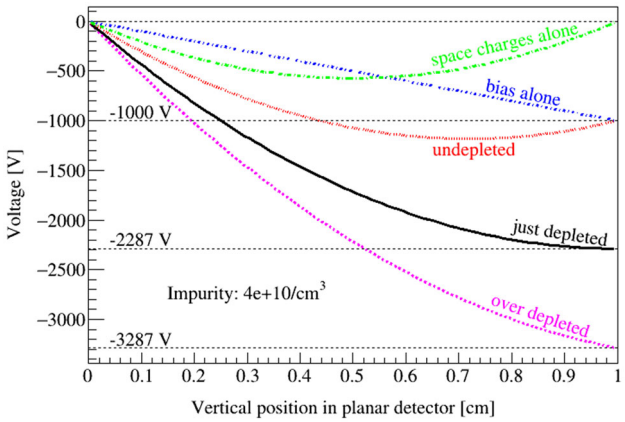

Fig. 12 Over depleted, just depleted, and undepleted voltage distributions in case of an ideal planar detector

What we are interested in here is to identify differences between an undepleted case and a depleted one. In this concrete example, an "undepleted" curve has $|V(x)| > |V_d|$ at some x. If we change the crystal from p-type to n and keep other configurations unchanged, the "space charge alone" curve would bend downward. More general criteria hence would be, an "undepleted" curve has $V(x)$ out of the range defined by boundary voltages, or $\mathrm{d}V/\mathrm{d}x$ changing its sign, at some x.

Assuming a certain bias voltage and a constant ρ over the whole volume, if the final answer is "undepleted" according to the criteria identified previously, we have to start over again assuming a higher bias. Obviously, the detector will certainly be depleted given an extremely high bias. However, in reality, it is hard to deliver a very high voltage without micro (or even major) discharges along the high voltage cable. Normally, the operation voltage is $\sim$ 1000 V over the depletion voltage.

The difference between an over depleted curve and a just depleted one, in this concrete example, is that

$$E(x = d) = -\mathrm{d}V(x = d)/\mathrm{d}x = 0 \tag{17}$$

for the latter, but $E(x = d) > 0$ for the former case.

In general, we have to do a search in between 0 and a large bias voltage for the "just depleted" case, where the electric field E on one of the boundaries is exactly zero. The calculation for this analytic example is very fast. Special treatment has to be taken in multi-dimensional numerical calculations to avoid expensive computations (see Sect. 5.3).

4.5 Impurity requirement

Given technical difficulties in delivering high voltages in a cryogenic environment, a low depletion voltage is generally preferred. It is a common practice to figure out the maximal net impurity concentration a crystal with certain dimensions must have to be depleted at or under a given voltage. In our previous example, the depletion requirement is $E(d) = 0$, which allows us to calculate C_2:

$$E(x = d) = \frac{\mathrm{d}V(d)}{\mathrm{d}x} = -\frac{\rho}{\epsilon}d + C_2 = 0 \Rightarrow C_2 = \frac{\rho}{\epsilon}d.$$

Insert the calculated C_2 and $C_1 = 0$ back to Eq. (7), we get the maximal allowed space charge concentration $\rho = 2V\epsilon/d^2$, where V is the given voltage.

Analytic solutions are not available for more complicated detector configurations. In that case, we need to make guesses on the impurity concentration or even profile, search for corresponding depletion voltages based on the method described in Sect. 5.3, and see if they go below the required voltage.

5 Numerical calculation

Even though many detector design concepts can be demonstrated with analytic solutions of highly symmetric detector configurations, numerical calculations are necessary for more advanced configurations that cannot be simplified to lower dimensional problems.

The first step of numeric calculation is to establish a grid within the detector volume, which consists of many tightly spaced points, some right on boundaries, others inside. The field values of a grid point can be determined by those of its immediate neighboring points. Their relations are dictated by Poisson's Equation in its numeric forms. Starting with the known values of the points on boundaries, the value of each point can be uniquely determined.

Configuring a grid that ensures an efficient and accurate calculation is an art by itself. For the sake of clarity in our discussion without losing generality, let's at first consider a section of a one dimensional (1D) grid around a point at x, as shown in Fig. 13.

Numerically, the second order derivative on the left side of Eq. (6) can be expressed as

$$\frac{\mathrm{d}^2V}{\mathrm{d}x^2} = \frac{(V_{i+1} - V_i)/\mathrm{d}x_+ - (V_i - V_{i-1})/\mathrm{d}x_-}{(\mathrm{d}x_+ + \mathrm{d}x_-)/2}, \tag{18}$$

where $\mathrm{d}x_\pm$ are the distances from the point at x to the previous and the next points as shown in Fig. 13. It is possible to involve more points in the calculation, such as the previous previous or next next points, but the basic idea is the same.

There are different ways to rearrange Eq. (6) based on Eq. (18), which lead to different methods to solve the problem. The two most common ones are the conjugate gradient method and the successive over-relaxation method.

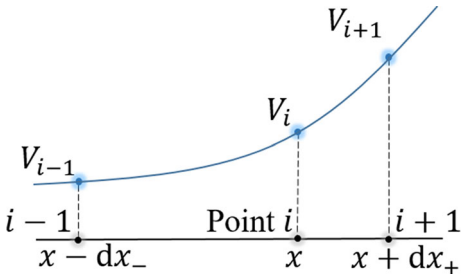

Fig. 13 A section of a 1D grid around a point at x

5.1 Conjugate gradient method

The conjugate gradient method starts by moving all known terms, such as the boundary voltages and terms containing $\rho(x)$, etc., to the right side of Eq. (6). Assuming n points in our 1D grid shown in Fig. 13, and $V(x) = V_i$, $\rho(x) = \rho_i$ are the values at the ith point, Eq. (6) becomes,

$$0\cdot V_1+\cdots+C_{i-1}V_{i-1}+C_iV_i+C_{i+1}V_{i+1}+\cdots 0\cdot V_{n-2} = K_i,$$

where C_i is the coefficient of V_i, K_i is the known term that contains ρ_i. We have such an equation for $n-2$ points, excluding the first and $n-1$ one, since $V_0 = 0$ and $V_{n-1} =$ the bias voltage are known and have to be included in K_i's. The $n-2$ linear equations can be collectively written as

$$CV = K, \tag{19}$$

where C is a $n-2$ by $n-2$ matrix, and V and K are vectors with $n-2$ elements. C is sparse, with at most 3 non-zero elements in each row. It is also symmetric and positive definite. A standard way to solve such a linear equation system is the *conjugate gradient algorithm* [32], which boils down to minimizing a quadratic function of V with the form $V^TCV/2 - K^TV$. There is a ROOT [33] macro included in GeFiCa to demonstrate the method. It works well when n is below 100, but becomes painfully slow for a large n.

5.2 Successive over-relaxation method

Another way to rearrange Eq. (6) is

$$V_i = \frac{\frac{\rho}{2\epsilon} + (V_{i+1}/\mathrm{d}x_+ + V_{i-1}/\mathrm{d}x_-)/(\mathrm{d}x_+ + \mathrm{d}x_-)}{(1/\mathrm{d}x_+ + 1/\mathrm{d}x_-)/(\mathrm{d}x_+ + \mathrm{d}x_-)}. \tag{20}$$

If $\rho = 0$ and $\mathrm{d}x_- = \mathrm{d}x_+$, it can be simplified to

$$V_i = (V_{i-1} + V_{i+1})/2, \tag{21}$$

where V_i is simply the average of its neighboring values. In both equations, V_i is calculated given V_{i-1} and V_{i+1}.

However, since V_{i-1} and V_{i+1} are also unknown (except for V_0 and V_{n-1}), we need to start the calculation with some initial values. One choice would be $V_0^{(0)} = V_1^{(0)} = \cdots V_{n-2}^{(0)} = 0$, and $V_{n-1}^{(0)} =$ the bias voltage, V_{bias}, where the superscript $^{(0)}$ indicates that these are the initial values of grid points.

Given these initial values, we can use Eq. (20) or (21) to update V_i. Use Eq. (21) in our calculation hereafter to simplify the demonstration, we have

$$V_i^{(j)} = [V_{i-1}^{(j-1)} + V_{i+1}^{(j-1)}]/2, \tag{22}$$

where j indexes the steps of updating. Since $V_{n-1}^{(j)} = V_{n-1}^{(j-1)} = V_{\mathrm{bias}}$, it pulls the value of its neighbor V_{n-2} up a bit after each updating, and V_{n-2} pulls up V_{n-3}, and so on and so forth. After many iterations, V_i becomes very close to its true value, the difference between the values of V_i in current and previous iteration becomes very small. We can use the following criterion to stop the iteration:

$$\left|\sum_i V_i^{(j)} - \sum_i V_i^{(j-1)}\right| < \text{a small value, e.g.,} 10^{-8}. \tag{23}$$

This is the so-called *successive relaxation* (SR) method.

To speed up the relaxation process, we can manually increase the difference of a value between two iterations by introducing a constant, $1 < F_R < 2$, in the following way:

$$V_i^{(j)} = V_i^{(j-1)} + F_R \times (V_i^{(j-0.5)} - V_i^{(j-1)}), \tag{24}$$

where, $V_i^{(j-0.5)}$, is the value updated by the original relaxation method, F_R is called the relaxation factor. This is why the method is called *successive over-relaxation* (SOR) method. The concept of SOR is depicted in Fig. 14, where the first a few steps of updating are shown for the last a few grid points in an ideal planar detector without any impurity. A carefully chosen relaxation factor can reduce the total number of iteration significantly, which is discussed in detail in Sect. 8.1.

To realize this idea in a program, one needs to create two arrays, one to hold the old voltage value, $V_i^{(j-1)}$, the other to hold the updated voltage, V_i^j, after the jth iteration. It is possible to simplify the program a bit by using only one array, which effectively changes Eq. (22) to

$$V_i^{(j)} = [V_{i-1}^{(j)} + V_{i+1}^{(j-1)}]/2, \tag{25}$$

where the already updated value at $i-1$, $V_{i-1}^{(j)}$ is used instead of the old one, $V_{i-1}^{(j-1)}$, to update the value at i. This is called *forward substitution*.

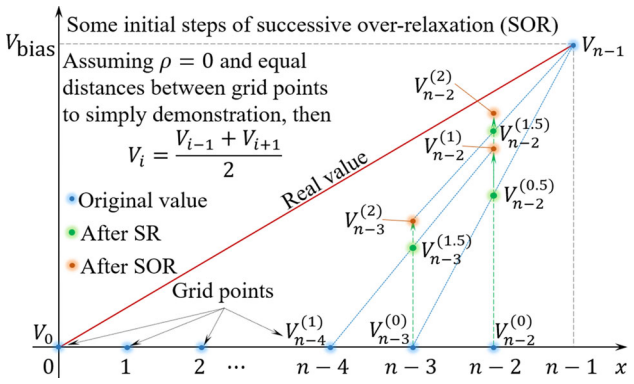

Fig. 14 Demonstration of how SOR increases the speed to approach the true value of a potential at a grid point i from an initial guess $V_i^{(0)}$

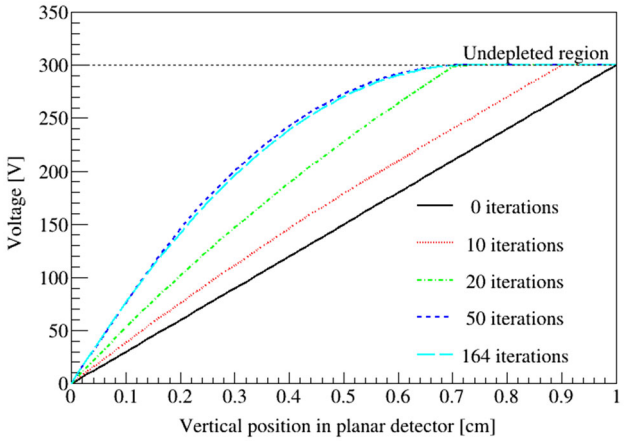

Fig. 15 Potential distribution of a planar detector with a $|V_{\text{bias}}| < |V_{\text{d}}|$ after some chosen numbers of SOR iterations. It shows the undepleted region as well as the converging process of SOR

The same method can be applied to multiple dimensional problems in various coordinate systems. The counterparts of Eq. (20) in those systems are summarized in "Appendix B".

5.3 Depletion voltage

The general method described in Sect. 4.4 applies to numeric calculations as well. We need to search for a V_{d} that just depletes the detector with the following procedure:

1. Pick up a V_{min}, say zero, and a V_{max}, say 10^6 V.
2. Assume a bias voltage, $V_{\text{bias}} \in (V_{\text{min}}, V_{\text{max}})$.
3. Run SOR until convergence.
4. Check if the detector is depleted. If not, replace V_{min} with V_{bias}; if yes, replace V_{max} with V_{bias}.
5. Repeat from step 2 until $V_{\text{max}} - V_{\text{min}} < 0.01$ volt.

The depletion voltage V_{d} is then $V_{\text{bias}} \approx V_{\text{max}} \approx V_{\text{min}}$ after a successful search.

One feature of the undepleted curve in Fig. 12 distinguishes it from the depleted ones; that is, the maximum or minimum of the potential is not on the boundaries of the detector. Inspired by this, the criterion of depletion in step 4 for numerical calculations can then be set as *none of the grid points has a potential that is larger or smaller than the value of any of its neighboring points*.

A potential drawback of the described method is that it may be time consuming if every new search needs to run an SOR. Fortunately, there is a way to avoid that. As demonstrated in Fig. 12, the total potential distribution, V_i, is a sum of the distribution due to impurity alone, V_i^{ρ}, and the one due to bias alone, V_i^{b}. Since $V_i^{\text{b}} = V_i^{\text{u}} \times V_{\text{bias}}$, where V_i^{u} represents the potential distribution due to unit voltage, 1 V, the total potential distribution can be calculated as

$$V_i = V_i^{\text{u}} \times V_{\text{bias}} + V_i^{\rho}. \tag{26}$$

If V_i^{u} and V_i^{ρ} are calculated using SOR before the described iteration, step 3 can be replaced by Eq. (26) instead of another SOR.

5.4 Undepleted region

When $|V_{\text{bias}}| < |V_{\text{d}}|$, some region of the detector is not depleted. Numerically, the undepleted region can be found by applying the following procedure to every grid point in the SOR process:

1. Calculate the potential of a grid point V_i using potentials of its immediate neighboring points.
2. Find the maximal and minimal potentials V_{max} and V_{min} of the immediate neighboring points.
3. Compare V_i with V_{max} and V_{min}. If $V_i < V_{\text{min}}$, it is set to be the same as V_{min}; if $V_i > V_{\text{max}}$, it is set to be V_{max}.

Figure 15 shows potential distributions of a planar detector with a $|V_{\text{bias}}| < |V_{\text{d}}|$ after some chosen numbers of SOR iterations. One can see how the undepleted region grows larger near one of the electrodes. Another interesting thing to notice is that it does not take many iterations for the potential to become very close to its final values. Most iterations after that are used to improve the accuracy in a few percent level.

As shown in Fig. 15, the undepleted region in a planar detector is adjacent to one of its electrodes. In the case of a point-contact detector, the undepleted region can stand alone somewhere in the center of the detector, away from any electrode. This is the so-called *pinch-off* effect, since the depleted region is "pinched off" from electrodes.

Since there is no electric field in the undepleted region to separate and drift electrons and holes generated by radiation

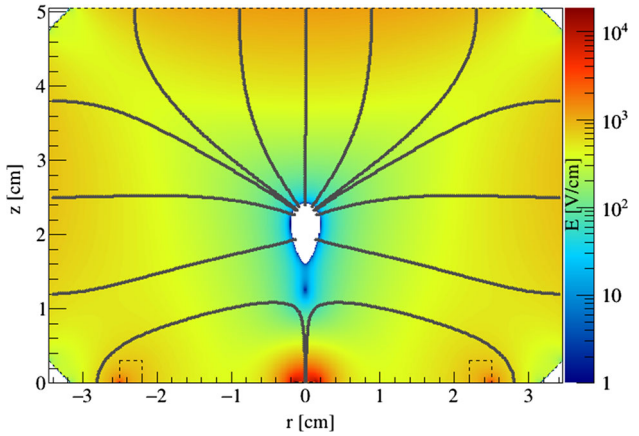

Fig. 16 The *pinch-off* effect demonstrated by the electric field as a colored contour in logarithm scale in a point-contact detector

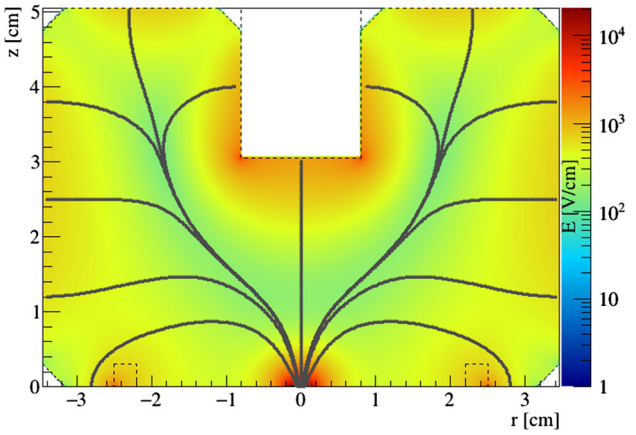

Fig. 18 Electric field distribution in an ICPC

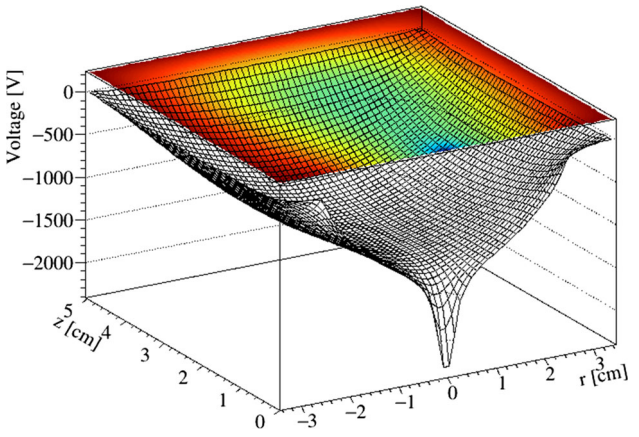

Fig. 17 Electric potential in the same detector as that in Fig. 16, shown as a colored contour on top and a 3D mesh at the bottom

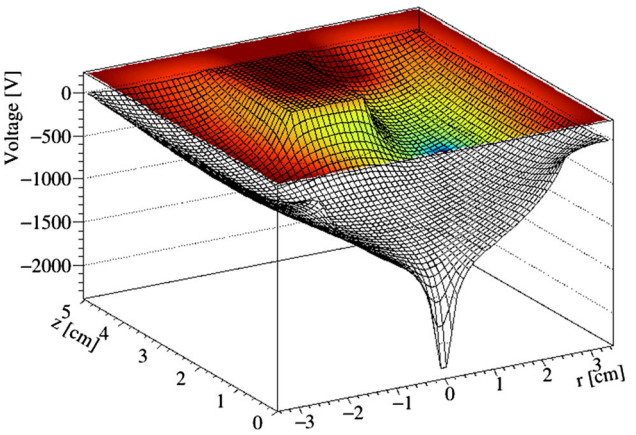

Fig. 19 Electric potential in the same ICPC as that in Fig. 18, shown as a colored contour on top and a 3D mesh at the bottom

interactions, the region is insensitive to radiation. Even if a pair of charge carriers are generated in the depleted region, one of them may drift to the undepleted region along the electric field and get stuck there instead of being collected by an electrode. It is hence worthwhile to reveal the existence of such an undepleted region through field calculation.

Figure 16 shows the electric field as a colored contour in logarithmic scale in a point-contact detector. The point contact is at the origin of the plot. The field value around the center of the detector is very close to zero, so the logarithm of them approaches negative infinity. They are color coded as white, which nicely visualizes the undepleted region that is *pinched off* from the point contact. Figure 17 shows the corresponding potential distribution in both a colored contour and a 3D mesh. Imagine the latter as a fancy water fountain. Water streams out of its top edges flow into a little pond in the middle of the fountain (the *pinched-off* undepleted region) instead of the sink (point contact). Only streams that are very close to the sink can flow directly into it instead of the pond.

This phenomenon seriously limits the size of a point-contact detector, since the electric field inside the detector becomes weaker when the size of the crystal becomes larger if the bias is not ramped up accordingly. To avoid this, one can bore a central hole from the opposite side of the point-contact, metallize its surface and keep it at the same bias as other surfaces. Such a detector is called a inverted-coaxial point-contact detector, or ICPC in short [34]. Figure 18 shows the electric field distribution in an ICPC as a color coded contour in logarithmic scale. Other configurations of this calculation are kept the same as the ones used to generate Fig. 16, including the crystal impurity level and the bias voltage. Figure 19 shows the corresponding potential distribution in both a colored contour and a 3D mesh. Using again the water fountain analogy to the 3D mesh, one can see clearly that the little pond (undepleted region) is successfully eliminated by raising part of the bottom of the fountain.

5.5 Electric field lines

The thick gray lines in Figs. 16 and 18 are estimated charge drift trajectories starting near the outer surface of the detectors. The procedure of the estimation can be illustrated in two dimensional Cartesian coordinates:

1. Linearly interpolate the electric field components E_x, E_y at a random starting point (x, y) using values at its four neighboring grid points.
2. Calculate the total electric field $E = \sqrt{E_x^2 + E_y^2}$ at the same point.
3. Calculate the propagation of a positive unit charge along x and y: $\mathrm{d}x = \mu E_x \mathrm{d}t$, $\mathrm{d}y = \mu E_y \mathrm{d}t$, where μ takes a value of 40, 000 cm^2/(volt·second), a number in between typical electron and hole drift mobilities in HPGe crystals [35–37], and $\mathrm{d}t$ takes a value of 10 ns.
4. $\mathrm{d}x$, $\mathrm{d}y$ are then further modified using equations $\mathrm{d}x = \mathrm{d}x \times weight$, $\mathrm{d}y = \mathrm{d}y \times weight$, where $weight = (5\ \mathrm{volt/mm})/E$, which is used to stretch $\mathrm{d}x$, $\mathrm{d}y$ in a weak field and shrink them in a stronger one.
5. The new position of the positive unit charge is then updated to $(x + \mathrm{d}x, y + \mathrm{d}y)$, which is saved in an object of the *FieldLine* class.
6. Repeat step 1 to 5 using the updated starting point coordinates until it moves out of the crystal or falls into an undepleted region.

Changing the positive unit charge to a negative one would let it propagate to the opposite direction.

Ignoring the influence of the germanium crystal structure, charge carriers drift roughly along electric field lines. The propagation path created this way can hence be regarded as a rough estimation of the field line.

It is interesting to see in Fig. 18 that the field lines merge in the middle of the detector and get collected at the point contact, just as streams flow down to a river in a valley, which is shown clearly with the 3D mesh in Fig. 19 and the greenish region in Fig. 18 if color-printed.

5.6 Boundaries in between grid points

Sometimes, the edges between the side and end surfaces of a cylindrical detector are tapered, shown as the small white triangular regions at the corners of the color contours in Figs. 16 and 18. A crystal boundary line hence can go in between grid points that are distributed along orthogonal lines, as shown in Fig. 20. Assuming a simple case, where the grid points are evenly spaced, the distances between them take fixed values, $\mathrm{d}x$ and $\mathrm{d}y$. The distances of a regular grid point to its previous and next neighbors equal to each other: $\mathrm{d}x_- = \mathrm{d}x_+ = \mathrm{d}x$. In case of a point near the boundary, such as (i, j) shown in

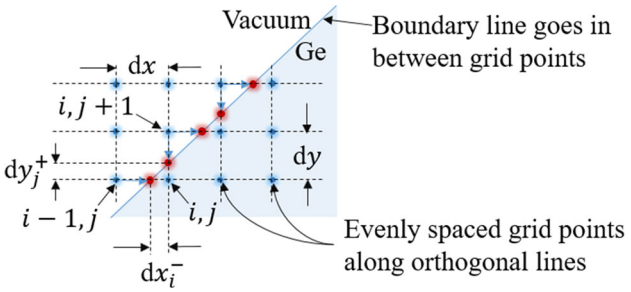

Fig. 20 Variable distances between grid points in case of a boundary line goes in between

Fig. 20, it is more precise to replace $\mathrm{d}x_-$ with $\mathrm{d}x_i^-$ when evaluating Eq. (18) along the x-axis, where $\mathrm{d}x_i^-$ is the distance to the boundary instead of the distance to the previous grid point as shown in Fig. 20. Similarly, $\mathrm{d}y_+$ should be replaced by $\mathrm{d}y_i^+$ for a more precise evaluation of Eq. (18) along the y-axis. This effectively moves nearby points on the vacuum side right to be on the boundary, shown as the red dots in Fig. 20. Such an operation can only be done if variable step lengths are allowed for individual grid points.

5.7 Weighting potential in segmented detectors

In addition to the real electric field, the so-called *weighting potential* is also of great interest, since it can be used to calculate the electric charges on an electrode induced by the drifting charge carriers inside a detector based on the Shockley–Ramo's Theorem [38–40]. It differs from a real potential in two ways. First, it is purely determined by its boundary conditions. The impurity concentration in a crystal should be regarded as zero in calculating the weighting potential. Second, the voltage on the interested electrode should be set to 1 volt, while the voltage on any other electrode should be set to zero in calculating the weighting potential. The weighting potential of an electrode is probably best demonstrated using a segmented HPGe detector. Figure 21 shows the cross section of a detector segmented evenly in six along the azimuthal direction. The weighting potential of one of the segment electrode is overlaid as a colored contour in a logarithm scale. The white circle in the middle indicates the core electrode of this cylindrical detector. The colored contour does not quit reach the bottom boundary, simply because the potential there is too close to zero to be color coded in a logarithm scale.

5.8 Capacitance

The capacitance of a HPGe detector C_d is of special interest due to at least two reasons. First, the electronic noise of a HPGe detector increases as C_d increases [39,41,42]. Second, C_d decreases as the detector bias voltage ramps up. The

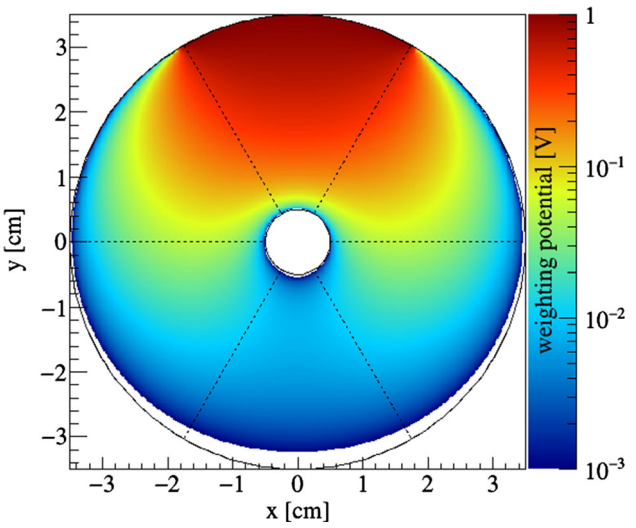

Fig. 21 Weighting potential distribution of a segmented detector (six evenly distributed segments along the azimuthal direction)

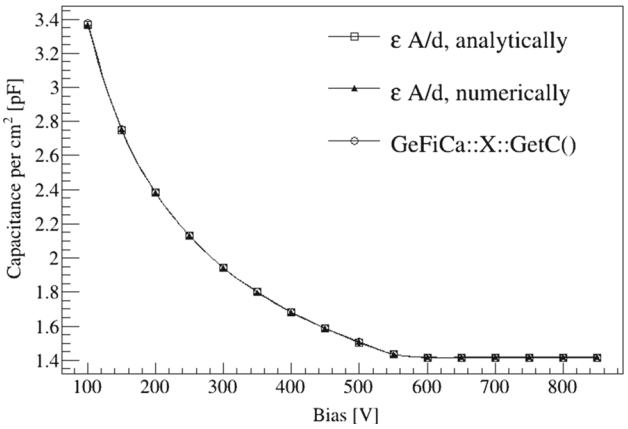

Fig. 22 Capacitance per unit area versus bias voltage of an ideal planar detector calculated in three different ways as detailed in the text. The agreement between each other verifies the correctness of the numerical calculation of the detector capacitance

reason becomes clear later in this section. This feature can be used to measure the depletion voltage, V_{d}. It can also be used to check if a detector operates normally during the ramping up of its bias voltage. It is therefore an important task of a field calculation package to calculate C_d given an arbitrary bias voltage.

For an ideal one dimensional planar detector,

$$C_d = \epsilon A/d, \tag{27}$$

where A is the area of an electrode of the detector and d is the thickness of the depleted region in the detector, which can be calculated as

$$d = \sqrt{2\epsilon V_{\mathrm{d}}/\rho}. \tag{28}$$

This relation can be derived from Eq. (7) with the boundary condition (15) and (17). C_d hence is anti-proportional to $\sqrt{V_{\mathrm{d}}}$, and decreases as V_{d} increases, until d becomes the thickness of the plan detector. After that, C_d stays at its minimum since d cannot increase anymore. The square data points in Fig. 22 are calculated using Eqs. (27) and (28) given individual bias values.

The depletion depth d can also be determined numerically using the method described in the previous section. C_d can be then calculated using Eq. (27). The results are the triangle data points in Fig. 22.

For a detector configuration as complex as a point-contact one, there is no analytic solution for C_d. The following numerical method is used in GeFiCa to calculate C_d. It is based on the fact that the energy stored in a charged capacitor U is equal to the overall work done W to move a total amount of charges Q to the electrodes against the electric field E caused by Q stored in the capacitor:

$$U = W. \tag{29}$$

Given an arbitrary amount of charges q already stored in a capacitor, the work done to increase it by an infinitesimal amount $\mathrm{d}q$ is

$$\mathrm{d}W = V_{\mathrm{bias}}\mathrm{d}q = (q/C_d)\mathrm{d}q. \tag{30}$$

Integrating it on both sides yields

$$W = \int_0^Q \frac{q}{C_d}dq = \frac{1}{C_d}\int_0^Q qdq = \frac{Q^2}{2C_d} = \frac{C_d V_{\mathrm{bias}}^2}{2}. \tag{31}$$

The relation $C_d = Q/V_{\mathrm{bias}}$ is used in the last step of the derivation to replace Q, an unknown variable, with C_d and V_{bias}.

On the other hand, since the electric field energy density is $\epsilon E^2/2$, U can be expressed as

$$U = \frac{1}{2}\epsilon \int_V E^2 \mathrm{d}\tau, \tag{32}$$

where $\mathrm{d}\tau$ is the volume integration element. For a planar detector with a constant impurity, the integral can be solved analytically as:

$$U = \frac{1}{2}\epsilon E^2 \int_V \mathrm{d}\tau = \epsilon V_{\mathrm{bias}}^2 A/(2d). \tag{33}$$

Replacing U and W in Eq. (29) with Eqs. (33) and (31), we derive Eq. (27).

The numerical version of Eq. (32) for an ideal planar detector in Cartesian coordinates is

$$U \approx \frac{1}{2}\epsilon \sum_{i=0}^{n} E_i^2 \mathrm{d}x_i A, \tag{34}$$

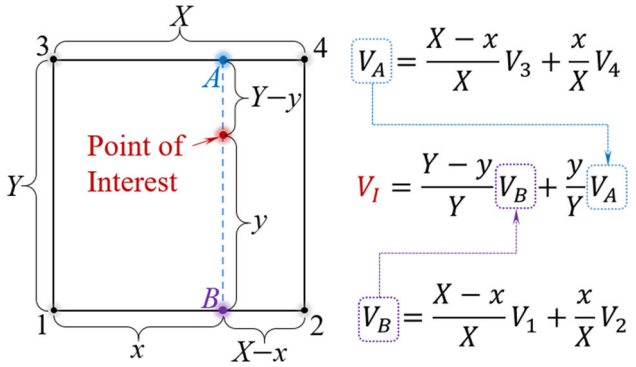

Fig. 23 Rectangular interpolation of the potential at the point of interest, V_I, using potentials at its nearest grid points, V_1, V_2, V_3 and V_4 in a 2D Cartesian grid

where i is the index of each grid point. Combining Eqs. (34), (31) and (29), C_d per unit area A can be calculated as

$$C_d/A = \epsilon \sum_{i=0}^{n} E_i^2 \mathrm{d}x_i / V_{\mathrm{bias}}^2. \tag{35}$$

This is implemented in function *GeFiCa::X::GetC()*. The results are shown as the circle data points in Fig. 22. The perfect agreement between all methods verifies two numerical calculations in GeFiCa: the finding of the undepleted region (or depleted region) and the calculation of C_d given an arbitrary V_{bias}.

It is worth noting that the electric field E here is only due to Q accumulated on the detector electrodes. It is different from the actual field in a depleted detector which is the combination of the fields from both Q and the space charge in the crystal.

For a point-contact detector in Cylindrical coordinates, the numerical version of Eq. (32) is

$$U \approx \frac{1}{2}\epsilon \sum_{i=0}^{n} E_i^2 r_i \mathrm{d}r_i \mathrm{d}z_i \int_0^{2\pi} \mathrm{d}\theta. \tag{36}$$

It is implemented in function *RhoZ::GetC()*.

5.9 Interpolation between grid points

A numeric calculation can only give the field values right at each grid point. Interpolations are needed to get the field values at a random point that may not coincide with any grid point. Figure 23 shows the equations to linearly interpolate the potential value at the point of interest that falls in between four points in a 2D Cartesian grid using the known potential values on those four points, V_1, V_2, V_3 and V_4, taking into account the distances between points, X, Y, x, y.

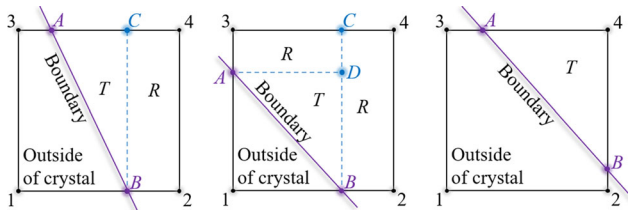

Fig. 24 Three ways that a boundary line goes through a unit grid square. Within the crystal, the point of interest can fall into either a triangular or a rectangular region marked as T or R

This method does not work for unit grid squares across crystal boundaries that are neither in parallel with nor perpendicular to grid lines, since those boundary lines can separate a square into irregular shapes, the interpolation of which can be complicated. There are three ways that such a boundary line can go through a unit grid square as shown in Fig. 24. Potentials at the crossing point, V_A and V_B, are equal to the bias applied to that boundary. Most of the time the field outside of the crystal is not of interest. Within the crystal, the point of interest can fall into either a triangular or a rectangular region marked as T or R, separated by blue dotted lines in Fig. 24. If it falls into an R region, the interpolation method shown in Fig. 23 can be used. If it falls into a T region, the triangular interpolation shown in Fig. 25 can be used.

The potential at the point of interest, V_I, in a T region can be calculated as the weighted sum of potentials at the grid points around, V_1, V_2 and V_3. The weights, W_1, W_2 and W_3 are the coordinates of the point of interest in the barycentric coordinates defined by the three grid points around. Since the Cartesian coordinates of all grid points and the point of interest are known, W_1, W_2 and W_3 can be calculated by transforming the Cartesian coordinates of the point of interest to the barycentric coordinates:

$$W_1 = \frac{(y_2 - y_3)(x - x_3) + (x_3 - x_2)(y - y_3)}{(y_2 - y_3)(x_1 - x_3) + (x_3 - x_2)(y_1 - y_3)},$$
$$W_2 = \frac{(y_3 - y_1)(x - x_3) + (x_1 - x_3)(y - y_3)}{(y_2 - y_3)(x_1 - x_3) + (x_3 - x_2)(y_1 - y_3)},$$
$$W_3 = 1 - W_1 - W_2,$$

where (x_1, y_1), (x_2, y_2) and (x_3, y_3) are the Cartesian coordinates of the grid points 1, 2 and 3 shown in Fig. 25, (x, y) are the Cartesian coordinates of the point of interest.

In case of the vector field E, interpolations are done separately for individual components to get E_x, E_y at the point of interest. The total E is then calculated as $\sqrt{E_x^2 + E_y^2}$.

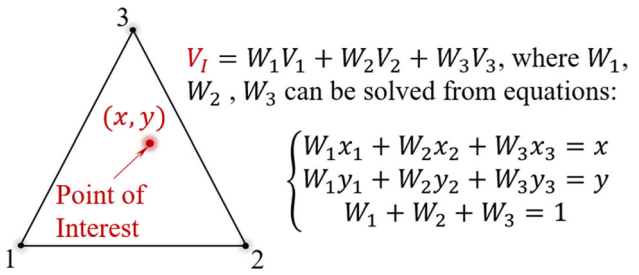

Fig. 25 Triangular interpolation of the potential at the point of interest, V_I, using potentials at its nearest 3 grid points, V_1, V_2, V_3

6 Implementation

6.1 Coding convention

The coding convention is similar to that of ROOT [43]. For example,

- Classes and functions all start with capital letters. Word boundaries are indicated by *CamelCase*.
- Classes names are all nouns.
- Function names are all verbs.
- Private member variables all start with letter *f*.
- Boolean variables/functions start with *Is/Are*.
- Indentation is made by three spaces instead of a hard tab to ensure the same appearance of the codes in different editors.

The following exceptions are used to increase the readability of the codes to the user:

- Class names do not have prefix letters, such as *T* in ROOT. Instead, the name space *GeFiCa* is used to avoid name collision should GeFiCa be used together with other libraries.
- Configurable member variables are made public to avoid trivial getters and setters. Their first letters are capitalized. Unlike private member variables, they do not have letter *f* prefixed.

6.2 Class structure

As shown in Fig. 26, most of the GeFiCa classes belong to two categories: *grid* and *detector*. Those that are derived from class *Grid* are used to describe grid setups. Those that are derived from class *Detector* are used to describe detector configurations. The *Grid* class inherits a set of arrays from the *Points* class to describe variables associated with individual grid points, such as coordinates and field values. Names of its derived classes indicate the dimension and coordinates used to construct the grid. For example, *X* is used for a one dimensional grid in Cartesian coordinates, *RhoZ* is used for a two dimensional grid in cylindrical coordinates. The *Detector* class inherits impurity setup from the *Crystal* class. Its derived classes, such as *PointContact*, inherit from it the common detector setups, such as bias voltages. A grid class can get boundary conditions and the impurity distribution from a corresponding detector class through a virtual function interface defined in the *Grid* class:

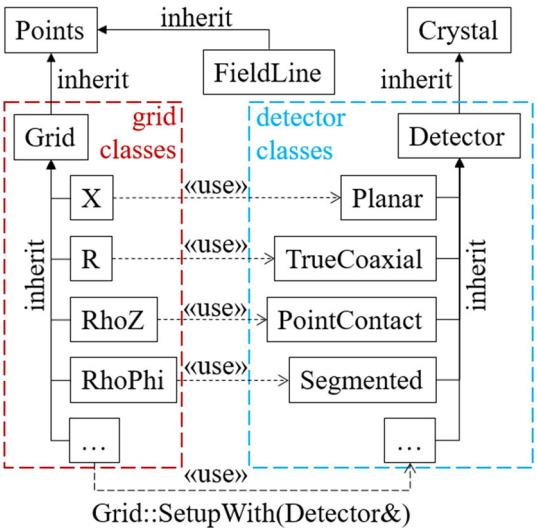

Fig. 26 Relation between GeFiCa classes

```
virtual void Grid::SetupWith(Detector&);
```

This is demonstrated in the following code snippet:

```
RhoZ grid; //create 2D Cylindrical grid
PointContact detector; //create detector
//setup grid with detector configuration
grid.SetupWith(detector);
```

The data flow can be the other way around, that is, a detector class gets grid setups from a grid class. However, since it is the grid that the SOR process updates instead of the detector configuration, this is a less natural choice. With the current data flow direction, the SOR can then be performed by simply calling

```
grid.SuccessiveOverRelax();
```

Another choice would be to combine the detector and grid classes. For example, instead of having both *PointContact* and *RhoZ*, we can create a single class called *PointContactRhoZ*. The advantage of this approaches is that there is no need to pass information from the latter to the former through some interface functions. The drawback is the lack of clarity, the same class object will be used for both detector configuration and grid operation. Considering the main purpose of GeFiCa is to demonstrate the logic, methods, and techniques for field calculation, we chose not to use this approach.

To its root, this is actually a question of to what extend we want to utilize the object-oriented (OO) coding style. Think about two extreme cases. First, we can write everything in a single main function. Second, we can create a class for each individual functionality, such as the impurity profile and the bias voltage. The first approach relies on careful documentation to clarify its internal logic. The second introduces many trivial interfaces to pass information between classes. A balanced approach in between is adopted for GeFiCa.

6.3 Data structure and I/O

Minimally, two float numbers are needed for each point in a one dimensional grid with a fixed interval between its points: one for the spacial coordinate and another for the electric field potential. The number of points in a grid must be changeable according to the dimension of a detector and the precision of a calculation. This demands the use of arrays that can change in size to store variables of individual points. Even though a *float* number is precise enough to hold the final result of a numerical calculation, a *double* is preferred to preserve precision during iterations of a SOR process. A standard C++ *vector<double>* is used in GeFiCa for each variable to provide enough precision and flexibility.

Given a grid with variable step sizes as shown in Fig. 13, one more variable is needed for each point to store the distance to its next neighbor, $\mathrm{d}x_i^+$. The distance to its previous neighbor is saved as $\mathrm{d}x_{i-1}^+$ in its previous point. In GeFiCa, however, the distances to both the previous and next neighbors, $\mathrm{d}x^-$ and $\mathrm{d}x^+$, are saved, since they may be different for some of the boundary points as detailed in Sect. 5.6. This certainly creates redundancy in storage for points away from boundaries. However, since output files of GeFiCa are saved in ROOT format instead of plain ASCII, such redundancy does not increase their sizes much due to the *gzip* compression algorithm used in ROOT to save equal-value variables only once.

In principle, electric field values can be calculated from the potential using Eq. (4) when needed. However, given their frequent usage, their values on each point are calculated and saved in GeFiCa after the SOR calculation for the potential.

For a three dimensional grid with a variable step size, 14 *std::vector*s with a *double* precision are needed in total to save three coordinates, 3×2 distances to previous and next points, one potential, one total electric field and its three components. They are public member variables in the class *Points* inherited by all *grid* classes shown in Fig. 26, including those representing lower dimensional grids, where variables for higher dimensions are not used at all. Since the C++ *vector* does not allocate memory if it has no element, there is no penalty in storage size in this solution. An alternative is to create *Points1D*, *Points2D*, *Point3D*, and consequently, *Grid1D*, *Grid2D*, *Grid3D* for various dimensions. This complicates the overall class structure unnecessarily, hence is not used in GeFiCa.

A few more *vector*s are added in the *Grid* class to record space charge densities in individual grid points, as well as flags to tell whether a point is in or out of a crystal, and whether it is in or out of the depleted region.

As described in Sect. 5.5, an electric field line can be saved in a series of points with variable distances between them. That is why the class *FieldLine* inherits the data structure from *Points*, as shown in Fig. 26.

The *Grid* and the *Detector* classes are both daughters of the *TNamed* class in ROOT, which inherits the capability to stream its data members for I/O from the *TObject* class in ROOT. Consequently, all concrete grid and detector classes can be directly saved into a standard ROOT file using one line of C++ as shown in the following code snippet:

```
TFile file("ICPC.root","recreate");
detector.Write(); // save config.
grid.Write(); // save grid
file.Close();
```

As described previously, repeated numbers in the ROOT file are compressed to save storage space. Detailed benchmark of the file size can be found in Sect. 8.2. After opening the ROOT file in a ROOT interactive session, one can use the Cling meta command *.ls* to list the saved objects:

```
root ICPC.root
root[].ls
TFile** ICPC.root
TFile*  ICPC.root
KEY: GeFiCa::PointContact pc;1 detector
KEY: GeFiCa::RhoZ rhoz;1 2D grid...
```

and directly use the loaded objects (*rhoz* and *pc*) to investigate and visualize the field and the detector:

```
root[] TTree *t = rhoz-> GetTree()
root[] t-> Draw("c2:c1:v","","colz")
root[] pc-> Draw()
```

The first line creates a *TTree* object *t* out of the saved field values in the *rhoz* object. The second line draws the potential, *v*, on the first (*y*-axis) and second (*x*-axis) coordinates, *c1* and *c2*, as a colored contour along *z*-axis (the "colz" option), as shown in Fig. 27.

The price to pay for all these convenience is that the objects saved in the ROOT file can only be loaded without warning message when the compiled GeFiCa library can be found and automatically loaded by ROOT. The way to realize this is detailed in Sect. 6.6.

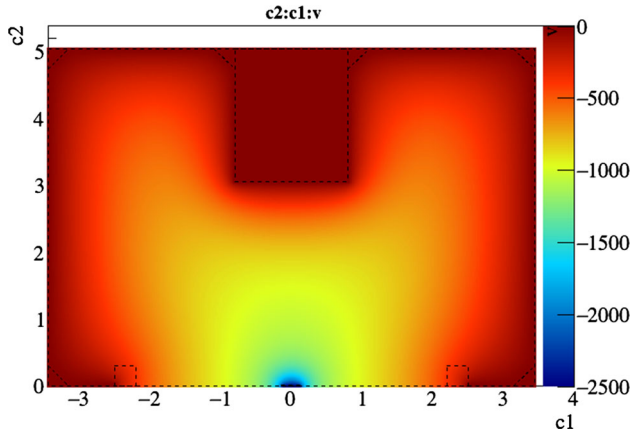

Fig. 27 The color contour of the potential field of an ICPC detector drawn with *TTree::Draw("c2:c1:v","","colz")*

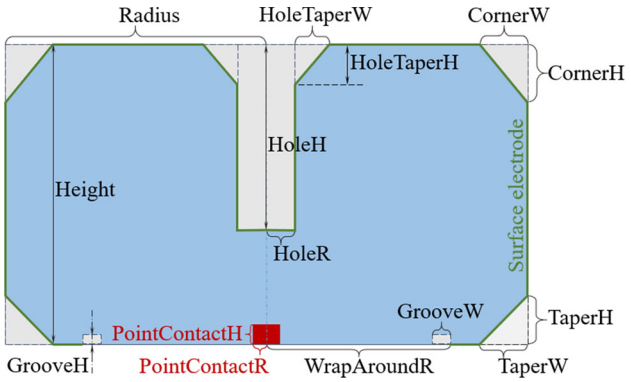

Fig. 28 Cross section along z-axis of an inverted coaxial point-contact HPGe detector, and parameters describing its dimensions. To shorten the names, width is represented as a single capital case "W", height "H", and radius "R"

6.4 Detector configurations

Two pieces of information are needed for electric field calculation: first, boundary conditions, and second, the space charge distribution.

Boundary conditions can be set through the detector geometry and voltages on electrodes. Take the previously defined *PointContact* detector as an example, its basic dimensions can be set as

```
detector.Radius = 3.45*cm;
detector.Height = 5.05*cm;
detector.PointContact R = 1.4*mm;
detector.PointContact H = 0.1*mm;
```

A full list of geometry parameters that can be set for a *PointContact* detector is shown in Fig. 28. Its bias voltages can be set as an array:

```
// point-contact voltage
detector.Bias[0] = -2.5*kV;
// surface contact voltage
detector.Bias[1] = 0*volt;
```

In case of a segmented detector, the bias voltage array can have more than two elements. The index of an element can be kept the same as the corresponding segment identification number.

As described in detail in Sect. 2, it is reasonable to use a first-order polynomial to approximate the space charge distribution in a HPGe crystal. With this simplification, we just need to specify the impurities at the top and the bottom of a crystal given by the manufacturer. For example,

```
detector.BottomImpurity=3e9/cm3;
detector.TopImpurity=7e9/cm3;
```

The impurity level at a specific axial position is interpolated in GeFiCa based on these two numbers. In case of a small crystal, the impurity can be regarded as a constant. Its average impurity can be set as

```
detector.SetAverageImpurity(3e9/cm3);
```

6.5 Units and constants

We have seen in previous code snippets that an input parameter in GeFiCa is a product of a number and a unit. Common units and constants for field calculation, together with their conversion rules, are defined in GeFiCa/src/Units.h. The following is a snippet of the file:

```
namespace GeFiCa {
  static const double C=1; // Coulomb
  static const double cm=1;
  static const double cm3=cm*cm*cm;
  static const double mm=0.1*cm;
  static const double volt=1;
  static const double kV=1000*volt;
  // vacuum permittivity [C/volt/cm]
  static const double epsilon0
  = 8.854187817e-14*C/volt/cm;
  // dielectric constant of Ge
  static const double epsilonR=16;
}
```

The advantage of this unit system is threefold. First, the code is self-explainable, there is no ambiguity in the unit of an input value. Second, the user has freedom to choose units, such as "mm" instead of "cm", or "kV" instead of "volt". Otherwise, he or she has to use the set of units used for internal calculation. Third, since the unit conversion rules are pre-defined, there is no need to worry about them when using input parameters for internal calculations. The programmer can focus on the logic instead of unit conversion. This way of handling units is adopted from Geant4 [22–24]. Most of

the units and constants have been defined in Geant4 already. However, since only a small subset of the units are useful for field calculation, they are re-defined in GeFiCa to avoid unnecessary dependence on Geant4.

6.6 Compilation and installation

GeFiCa relies on ROOT to realize C++ and Python scripting, efficient I/O and plotting. It has to be compiled against ROOT libraries. This is achieved through a simple Makefile that uses the root−config executable available from any successfully installed ROOT package to get the location of ROOT libraries and necessary compilation flags. The compilation process is as simple as

```
cd /path/to/GeFiCa/src && make
```

After a successful compilation a shared C++ library, libGeFiCa.so, can be found in /path/to/GeFiCa/src. Once its location is added to the LD_LIBRARY_PATH environment variable (or DYLD_LIBRARY_PATH in MacOS), the library can be automatically loaded only when needed as any other ROOT libraries in ROOT interactive sessions or scripts thanks to the *rootmap* and *pcm* files [44] generated by the *make* process in the same directory as the library.

6.7 Supported OS

Since ROOT is available in the three common operating systems, Linux, Windows and MacOS, in principle, GeFiCa should be able to be compiled in all of them as well. However, since GeFiCa relies on a simple Makefile to compile, it cannot be directly compiled through the native Windows compilation system. Instead, it can be compiled in a Windows Subsystem for Linux (WSL). To date, GeFiCa has been compiled successfully in CentOS 6 and 7, MacOS 10.12, 10.13, 10.14, and Ubuntu 18.04 as a WSL.

6.8 Code accessibility

The codes of GeFiCa are hosted online at GitHub [45]: https://github.com/jintonic/gefica. They can be downloaded directly from the web page or through *git*. GeFiCa is release under the MIT License [46]. It can be freely used without any warranty as long as the license is distributed along with it.

6.9 Code documentation

A git branch *gh-pages* is used to host the homepage code for GeFiCa. The homepage is available under a customized domain name: http://physino.xyz/gefica. It lists three main resources about GeFiCa that one can get help from: the GeFiCa repository page hosted on GitHub, the code documentation hosted on https://codedocs.xyz, and the user manual hosted on https://readthedocs.org.

There is a *README.md* file in each directory in GeFiCa to explain the contents of the directory written in *GitHub Flavored Markdown* format [47]. They are rendered to web pages automatically in GitHub. A user can quickly get help with or without the source code.

Explanations of GeFiCa classes and variables are embedded in the source code as C++ comments using the Doxygen [48] convention. They can be rendered by Doxygen into nicely formatted documentations locally or on https://codedocs.xyz. The online version is updated automatically once new codes are pushed to the GitHub repository.

The user manual is written in *restructured text* format and can be rendered to web pages locally or on https://readthedocs.org. The online version is updated automatically once new documentation is pushed to the GitHub repository.

In addition to these, example codes are shipped with GeFiCa as ROOT macros as described in detail in the next section to demonstrate the usage of individual GeFiCa classes.

6.10 Macros and scripts

A modern C++ interpreter, cling [49], has been created and adopted as the back-end of the interactive session of ROOT [33] since the version 6 of it. A user can run C++ snippets, sometimes called ROOT macros or scripts, interactively in cling without writing and compiling the "main" function. With immediate feedback after the execution of each line of a script, a user can learn and experiment a new C++ class, a function, or simply a syntax easily. To fulfill its educational purpose, GeFiCa is compiled as a ROOT library. All snippets in previous sections demonstrating the configuration of a detector or the operation of a grid can be run as they are in cling.

ROOT also provides a Python extension module, PyROOT, that allows the user to interact with any ROOT class from the Python interpreter. For users who prefer the Python interpreter to cling, they can call GeFiCa classes with Python syntax directly in the standard Python interpreter.

It is worth noting that cling comes with a Jupyter [50] kernel, which makes it possible to run GeFiCa scripts in a Jupyter notebook with either C++ or python syntax.

All concrete grid and detector classes in GeFiCa inherit the capability to inspect themselves from the *TObject* class in ROOT. Some standard functions in TObject, such as *Dump()*, can be used to check the default or user-specified configurations of a grid or detector object, as shown in Listing 1. The first column of the output are the member variables of the *GeFiCa::X* class. The second are their current values. The last are explanations of those variable. These explanations

are written as C++ comments after the member variables. They can be parsed by both Doxygen and ROOT to generate code documentation in various formats and contexts.

Macros are organized in sub-folders in *GeFiCa/examples/* to demonstrate the usage of GeFiCa classes. The *planar/*, *trueCoaxial/*, *hemispherical/*, *pointContact/*, and *segmented/* folders are used to show how to configure specific types of HPGe detectors and then calculate the fields in them. The *analytic/* and the *fenics/* folders contains macros that are independent of the *GeFiCa* libraries. The macros in the former demonstrate how to calculate and visualize the field distribution in simple HPGe detectors using ROOT. The latter shows Python codes to calculate and visualize the field distribution in a simplified point-contact geometry using FEniCS [19]. All field distributions shown in this work are generated using these macros. A user can learn the topics by at first running these macros to reproduce plots in this work, and then modifying them to meet his/her own needs.

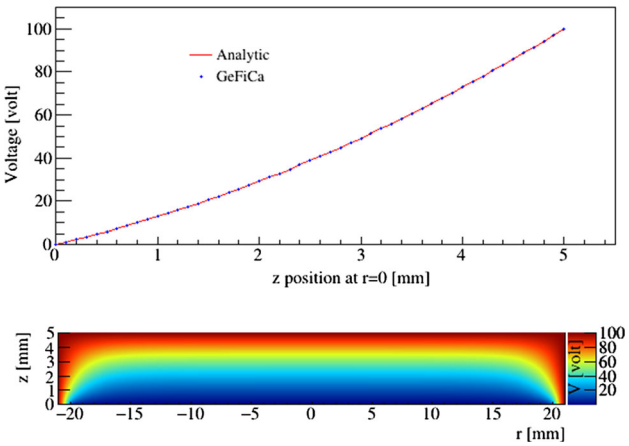

Fig. 29 Top: Comparison of the electric potential calculated numerically in a pancake-like "point-contact" detector with the analytic solution of a planar detector that has the same thickness (or height) and impurity concentration. Bottom: The electric potential distribution calculated numerically in the pancake-like detector, the "point-contact" of which is artificially enlarged to cover almost the entire bottom end surface

7 Code verification

A common way to verify the saneness of a complex theory in physics is to consider extreme conditions, under which the theory can be simplified and compared to predictions based on common sense. Take the field in a point-contact detector as an example, there are two extreme cases where the field in certain part of the detector can be regarded as the same as that in a planar or a true-coaxial detector. This makes it possible to compare the numeric calculation of a point-contact detector field directly with analytic solutions.

7.1 Comparison with analytic solutions

In the first extreme case we consider a point-contact detector that takes a pancake-like shape, that is, its thickness is much smaller than its diameter. Furthermore, its "point-contact" covers almost the entire bottom end surface. The electric potential in such a detector is shown in the bottom plot in Fig. 29. At the radial center of the detector, the field is essentially the same as that in a planar detector that has the same thickness and impurity concentration. In the top plot in Fig. 29 the analytic solution of such a planar detector is overlaid on top of the numerical result of the pancake-like "point-contact" detector along the z (axial) positions at r (radial position) $= 0$.

In the second case let us consider a point-contact detector that looks like a thin stick, that is, its diameter is much smaller than its height. Furthermore, let's make its "point-contact" as deep into the crystal as possible. The potential distribution in such a detector calculated numerically is shown in the right plot in Fig. 30. Far away from the top end of the detector, the field is essentially the same as that in a true-coaxial detector that has the same radius and impurity concentration. In the left plot in Fig. 30 the analytic solution of such a true-coaxial detector is overlaid on top of the numerical result of the thin-stick-like "point-contact" detector along r (radial position) at z (axial position) $= 5$ mm above the bottom surface.

7.2 Comparison with fieldgen

Even though the perfect matches between the analytic solutions and the numerical results in both cases are convincing evidences of the correctness of the numerical calculation implemented in GeFiCa, it is worth noting that a constant impurity concentration throughout the entire crystal is assumed to make the analytic solutions possible. In case of an arbitrary impurity distribution, no simple analytic solution is available, the numerical calculation in GeFiCa is compared to that of fieldgen [1,14], a thoroughly examined and widely accepted package in the field, given identical point-contact detector configurations.

The biggest difference between GeFiCa and fieldgen in the aspect of numerical calculation is probably the setup of grid points. In case of fieldgen, the grid points along the radial direction, r, of a detector start from $r = 0$ and end at $r =$ the radius of the detector. In case of GeFiCa, the grid points are in the range of [−radius, +radius] and there is no grid point at $r = 0$ to avoid setting artificial boundary conditions at $r = 0$. Due to this difference, there is no one-to-one correspondence between a grid point in GeFiCa and that in fieldgen. In order to make a point-to-point comparison, linear interpolation is used to get the total electric field strength at a fieldgen point from two nearby GeFiCa points, the interpolated value is then compared to the fieldgen value at the same point. Their

Listing 1 A truncated ROOT interactive session displaying the contents of an object of the *GeFiCa::X* class.

```
root [] GeFiCa::X x
(GeFiCa::X &) Name: x Title: 1D Cartesian coordinate
root [] x.Dump()
==> Dumping object at: 0x00007f76e5d80150, name=x, class=GeFiCa::X
Src                 ->7f76e5d802a8      -(net impurity concentration)x|Qe|/epsilon
N1                  101                 number of points along the 1st coordinate
N2                  0                   number of points along the 2nd coordinate
N3                  0                   number of points along the 3rd coordinate
MaxIterations       5000                maximal iterations of SOR to be performed
RelaxationFactor    1.95                within (0,2), used to speed up convergence
Tolerance           1e-07               SOR stops when error<Tolerance
...
```

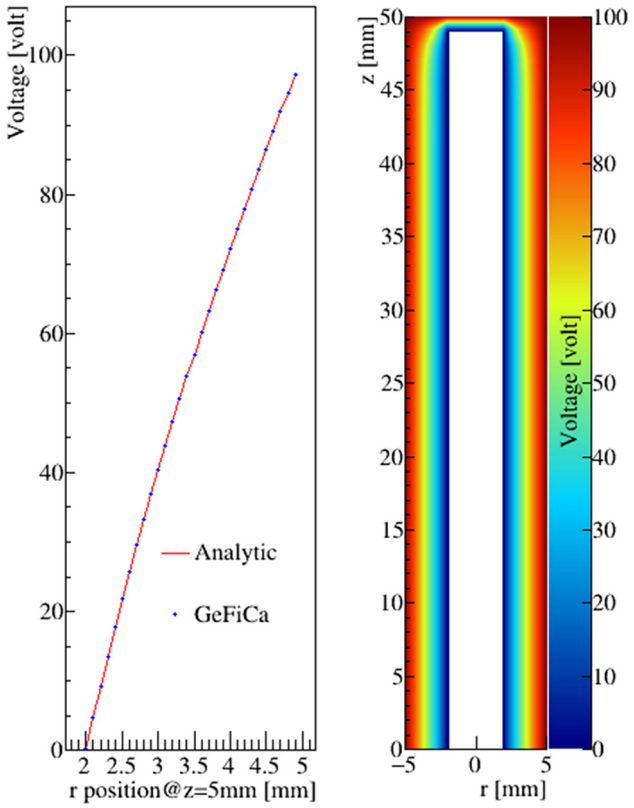

Fig. 30 Left: Comparison of the electric potential calculated numerically in a thin-stick-like "point-contact" detector with the analytic solution of a true-coaxial detector that has the same radius and impurity concentration. Right: The electric potential distribution calculated numerically in the thin-stick-like detector, the "point-contact" of which is artificially prolonged along almost the entire height of the crystal

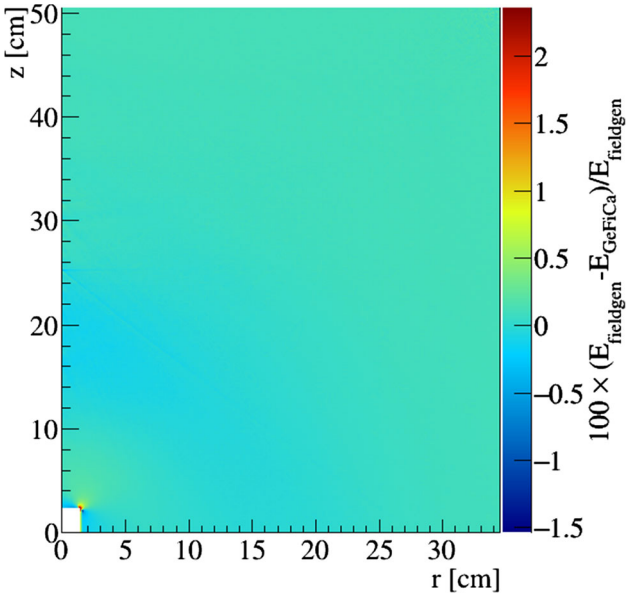

Fig. 31 Relative difference between the electric potential distributions calculated using fieldgen and GeFiCa for an identical point-contact detector configuration

relative difference in percentage is shown as colored contour in Fig. 31.

The largest difference is about 8.5% at the top right corner of the point-contact. This point is removed from Fig. 31 so that subtle differences between fieldgen and GeFiCa are more visible in the figure. The second largest difference is about 2.5% at an adjacent point, shown as the red spot in Fig. 31. The difference quickly falls below 0.1% only a few points away from the corner, which translates to about one millimeter in length given the 0.1 mm distance between grid points. Such difference is most probably due to different treatments in fieldgen and GeFiCa on grid points near boundaries.

Fortunately, the difference is of little importance in practice since there is no such sharp corner inside any detector in reality. Predictions of GeFiCa and fieldgen in the bulk of the detector are essentially identical.

8 Performance

8.1 Relaxation factor

As described in Sect. 5.2, the number of iterations needed for a successive relaxation process to converge can be reduced

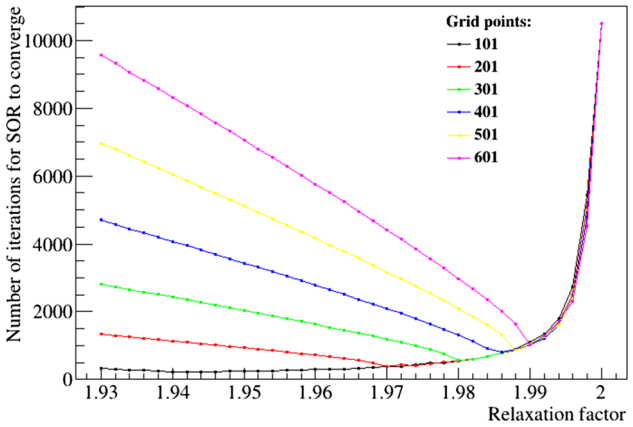

Fig. 32 Number of SOR iterations versus relaxation factor

by introducing a relaxation factor in between [1, 2). Figure 32 shows the number of iterations for a successive over-relaxation (SOR) process to converge as a function of the relaxation factor. Each data point in the figure represents the result from a numerical calculation of the field in an ideal planar detector. Data points that are connected by lines in between are from calculations sharing the same number of grid points.

A common trend shared by all the lines is that there is a point where the number of iterations is minimized. That is where GeFiCa reaches its best performance. As the number of grid points increase from 101 to 601, the relaxation factor corresponding to the minimal iteration numbers increases from around 1.94 to 1.99. The default value of the relaxation factor is set to 1.95 in GeFiCa. A user can change it using the following line of code if desired.

```
grid.RelaxationFactor=1.99;
```

The gain in performance by selecting an appropriate relaxation factor becomes more prominent when the number of grid points becomes larger. Take the up most curve in Fig. 32 as an example, which corresponds to calculations done with the finest grid, when the relaxation factor changes by only 0.06 from 1.93 to 1.99, the number of iterations reduces from about 10,000 to less than 2000. While the lowest curve in Fig. 32 is almost flat around 1.94, that is, the relaxation factor cannot help much to gain speed for calculations with a very coarse grid. This is not a problem since those calculations are fast already.

As pointed out in Ref. [51], it is often hard or even not possible to compute in advance the optimized relaxation factor for a fast convergence. A heuristic estimation, $F_R = 2 - \mathcal{O}(\mathrm{d}x)$, seems to work as an initial guess for the two dimensional calculation for point-contact detectors, where $\mathrm{d}x$ is the step length between two adjacent grid points.

After every 100 iterations, GeFiCa prints the overall difference (error) of potentials at all grid points between current and previous iterations. When the error is smaller than a target tolerance (1×10^{-7} V by default), the SOR is regarded as converged, the calculation stops there, and the CPU time used for the calculation is printed out on screen as shown in the terminal output below:

```
root [0] .x calculateFields.cc
Processing calculateFields.cc...
Info in < GeFiCa::RhoZ::SuccessiveOverRelax>:
Start...
    0 steps, error: 1.0e+00 (tolerance: 1e-07)
  100 steps, error: 4.8e-03 (tolerance: 1e-07)
  200 steps, error: 2.7e-03 (tolerance: 1e-07)
    .
    .
    .
2000 steps, error: 1.0e-07 (tolerance: 1e-07)
2004 steps, error: 1.0e-07 (tolerance: 1e-07)
Info in <GeFiCa::RhoZ::SuccessiveOverRelax>:
CPU time: 23.2 s
```

This terminal output is associated with the calculation used to generate Fig. 31. The overall number of grid points is 349,140. The CPU time used for the calculation is about 23 s in a Linux server with an Intel Xeon Gold 5118 CPU at 1 GHz. The relaxation factor chosen for this calculation is 1.994.

8.2 Output file size

The output of fieldgen used to generate Fig. 31 is saved as a simple ASCII file that is 8.1 MB in size. The detector configuration is saved as a short header of the file. The rest of the file are six columns of values of the grid point positions (radial and axial), the voltage, the overall electric field strength, and its radial and axial components.

As described in Sect. 6.3, the detector and grid objects in GeFiCa can be directly saved in a standard ROOT file. Its contents can be printed and visualized in a ROOT interactive session as demonstrated in the code snippets in Sects. 6.3 and 6.10. In addition to the information saved in a fieldgen output, a GeFiCa output also contains the intervals between grid points, flags indicating whether a point is depleted or not, etc. It also contains about twice more grid points than fieldgen. In total, the amount of information saved in GeFiCa is about four times more than that saved in the fieldgen output. The size of the GeFiCa output ROOT file used to generate Fig. 31 is 9.2 MB, only slightly larger than that of the fieldgen output file, thanks to the *gzip* algorithm used to compress a ROOT file mentioned in Sect. 6.3.

9 Extendability and limitation

Let's take a realistic planar detector configuration shown in Fig. 33 as an example to demonstrate the procedure of extending GeFiCa for a new type of detector.

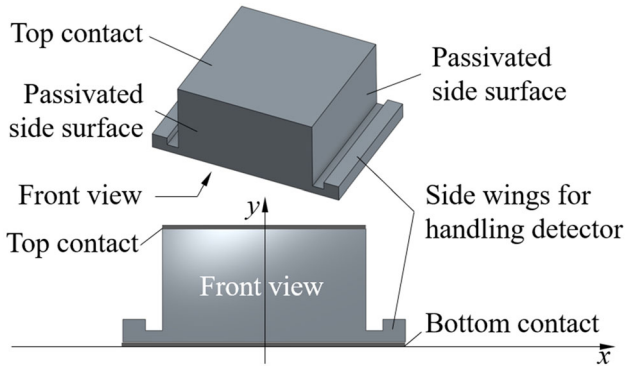

Fig. 33 Configuration of a realistic planar detector that has two side wings for the handing of the detector. The top and bottom surfaces are electrical contacts, the side surfaces are passivated

The top and bottom surfaces of the detector are covered with a thin layer of aluminium to form the electric contacts. All the side surfaces are covered with a thin layer of amorphous germanium for passivation purpose. The two side wings can be used for handling the detector without touching its sensitive surfaces [52]. Since they are thin compared to the overall thickness of the detector, the electric field distribution inside the detector can hence be approximated by that in an ideal 1D planar detector. However, if our intention is to study the influence of the thickness of the wings on the electric field, we need at least a 2D grid in Cartesian coordinates to perform the numerical calculation, which can be achieved with the following steps.

At first, a class called *XY* that represents the dimension and coordinates needs to be created. It inherits all member variables in its base class *Grid* that define the grid. Since the numerical expression of Poisson's equation (Eq. 5) depends on dimensions and coordinates used for the calculation, a protected virtual function, *void OverRelaxAt(size_t idx)*, in *Grid* needs to be overwritten in *XY*, which takes care of the updating of the field value at each grid point indexed by *idx*.

Secondly, a class called *TopHat* that describes the geometry of the detector needs to be created. It inherits the member variables that hold voltages values of all electrodes from its base class *Detector*. It also inherits the impurity distribution from the class, *crystal*. A public member function *void Draw()* in *Detector* needs to be overwritten in TopHat to visualize the geometry setup.

At last the public virtual function in *Grid* called *void SetupWith(Detector&)* needs to be overwritten in *XY*, which takes the boundary conditions and impurity distribution from *TopHat* to construct and initialize the grid for the calculation.

For completeness, a folder called *TopHat* is recommended to be created under *GeFiCa/examples*, which contains ROOT or Python scripts demonstrating the usage of *XY* and *TopHat*.

Given its extendability, there is no limitation on GeFiCa from the functionality point of view. From the education point of view, however, there is currently no function in GeFiCa demonstrating the adaptive grid configuration that automatically updates distances between grid points over iterations based on the strength of local electric field. Note that there is no fundamental limitation from GeFiCa inhibiting doing so, since there are separated member variables in the *grid* class to hold distances from a grid point to its neighbors in all directions. Practically, GeFiCa is already fast and precise enough with fixed step length for common HPGe configurations. This function can be added in if necessary.

10 Summary

The new educational program, GeFiCa, has been created to demonstrate analytic and numeric methods to calculate static electric fields and potentials in HPGe detectors. It is freely available from http://physino.xyz/gefica and can be installed in three major operating systems, Linux, MacOS and Windows, as a CERN ROOT [33] library extension. Powered by ROOT, GeFiCa allows its users to explore in detail the calculation procedure by executing C++ or Python code snippets in ROOT interactive sessions or Jupyter notebooks without compilation. Example code snippets are shipped together with the library to demonstrate calculations for common detector configurations, and to visualize the resulting field distributions in graphs or color contours. In addition to field calculations, GeFiCa offers functionalities to calculate the HPGe detector depletion voltage, undepleted region, capacitance, etc., that are not available from general-purpose field calculation programs, such as Maxwell3D and FEniCS. Compared to open projects that are also specialized in HPGe field calculation, such as fieldgen and SSD, etc., GeFiCa offers a ROOT-based C++ solution that is equally accurate and efficient, and shipped with a large amount of documentations and examples that are not readily available in others.

This article was written to provide an entry level review of methods and tools available at the moment, with the hope that its readers feel comfortable to make an educated choice of simulation tools best suited for the task at their hands.

Acknowledgements The authors thank David Radford at the Oak Ridge National Laboratory for his patient instruction in various aspects of the field calculation, Oliver Schulz at the Max-Planck-Institut für Physik for his introduction of the Julia language and the SSD package, Christopher Haufe and Anna Reine at the University of North Carolina at Chapel Hill for their instruction on how to use FEniCS to calculate fields in a point-contact detector. This work is supported by NSF award OIA-1738695 and OISE-1743790, and the Office of Research at the University of South Dakota. Computations supporting this project were performed on High Performance Computing systems at the University of South Dakota, funded by NSF award OAC-1626516.

Data Availability Statement This manuscript has no associated data or the data will not be deposited. [Authors' comment: Data used to create plots in this article can be generated using the program described in the article.]

Funded by SCOAP[3].

Appendix A: Poisson's equation in curvilinear coordinates

The Poisson's equation in spherical coordinates reads,

$$\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial V}{\partial r}\right)+\frac{1}{r^2\sin\theta}\frac{\partial V}{\partial\theta}\left(\sin\theta\frac{\partial V}{\partial\theta}\right)+\frac{1}{r^2\sin^2\theta}\frac{\partial V^2}{\partial^2\phi}=-\frac{\rho(r,\theta,\phi)}{\epsilon}, \tag{A.1}$$

where $r,\theta\in[0,\pi]$, $\phi\in[0,2\pi)$ are the radial distance from the origin, the polar angle and the azimuth angle, as defined in the left plot of Fig. 34.

The Poisson's equation in cylindrical coordinates reads,

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial V}{\partial r}\right)+\frac{1}{r^2}\frac{\partial V^2}{\partial^2\phi}+\frac{\partial V^2}{\partial^2 z}=-\frac{\rho(r,\phi,z)}{\epsilon}, \tag{A.2}$$

where $r,\phi\in[0,2\pi)$, z are the radial distance from the origin, the azimuth angle, and the height, as defined in the right plot of Fig. 34. A more commonly used symbol of the radial distance in the cylindrical coordinate system is ρ. However, r is used here instead of ρ to avoid being confused with the space charge density, which is denoted as ρ as well.

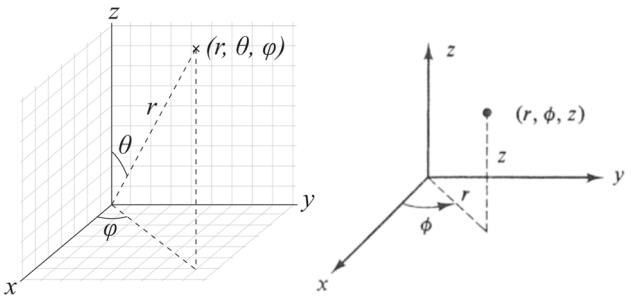

Fig. 34 Definitions of spherical coordinates (left) and cylindrical ones (right)

Appendix B: Iteration relations

In 3D Cartesian coordinates, the potential at a grid point after the ith successive relaxation iteration, V_{i+1}, can be expressed as

$$\begin{aligned}V_{i+1}=&\left\{\frac{\rho}{2\epsilon}+\left[\frac{V_i(x+dx_+)}{dx_+}+\frac{V_i(x+dx_-)}{dx_-}\right]\frac{1}{dx_++dx_-}\right.\\&+\left[\frac{V_i(y+dy_+)}{dy_+}+\frac{V_i(y+dy_-)}{dy_-}\right]\frac{1}{dy_++dy_-}\\&\left.+\left[\frac{V_i(z+dz_+)}{dz_+}+\frac{V_i(z+dz_-)}{dz_-}\right]\frac{1}{dz_++dz_-}\right\}\\&/\left[\left(\frac{1}{dx_+}+\frac{1}{dx_-}\right)\left(\frac{1}{dx_++dx_-}\right)\right.\\&+\left(\frac{1}{dy_+}+\frac{1}{dy_-}\right)\left(\frac{1}{dy_++dy_-}\right)\\&\left.+\left(\frac{1}{dz_+}+\frac{1}{dz_-}\right)\left(\frac{1}{dz_++dz_-}\right)\right],\end{aligned}$$

where x,y,z are coordinates, dx_+,dy_+,dz_+ are distances to the next grid points, dx_-,dy_-,dz_- distances to the previous.

To achieve this in a program, two arrays are needed as explained at the end of Sect. 5.2. In case of forward substitution, half of the points around x have already been updated before the calculation of $V_{i+1}(x)$, that is, half of the i in the equation above should be replaced by $i+1$. It is a bit of work to keep track of which points have been updated in the equation above. In a program, however, this is achieved automatically if only one array of V is used to hold values at all grid points. Since there is no need to keep track of which points have been updated in this case, one can safely drop the subscription $i, i+1$ in the equation above. This holds true for all equations hereafter.

In a 1D cylindrical coordinate, the potential at a grid point after the ith successive relaxation iteration, V_{i+1}, can be expressed as

$$\begin{aligned}V_{i+1}=&\frac{\rho}{\epsilon}\frac{\mathrm{d}r_+\mathrm{d}r_-}{2}\\&+\frac{V_i(r+\mathrm{d}r_+)-V_i(r-\mathrm{d}r_-)}{2r}\Big/\left(\frac{1}{\mathrm{d}r_+}+\frac{1}{\mathrm{d}r_-}\right)\\&+\left[\frac{V_i(r+\mathrm{d}r_+)}{\mathrm{d}r_+}+\frac{V_i(r-\mathrm{d}r_-)}{\mathrm{d}r_-}\right]\Big/\left(\frac{1}{\mathrm{d}r_+}+\frac{1}{\mathrm{d}r_-}\right),\end{aligned}$$

where r is the coordinate. dr_+ is the distance to the next grid point, dr_- the distance to the previous.

In 3D cylindrical coordinate, the potential at a grid point after the ith successive relaxation iteration, V_{i+1}, can be

expressed as

$$V_{i+1} = \left\{ \frac{\rho}{2\epsilon} + \frac{1}{r}\frac{V_i(r+\mathrm{d}r_+) - V_i(r+\mathrm{d}r_-)}{\mathrm{d}r_+ + \mathrm{d}r_-} + \left[\frac{V_i(r+\mathrm{d}r_+)}{\mathrm{d}r_+} + \frac{V_i(r+\mathrm{d}r_-)}{\mathrm{d}r_-}\right]\frac{1}{\mathrm{d}r_+ + \mathrm{d}r_-} + \left[\frac{V_i(\theta+d\theta_+)}{d\theta_+} + \frac{V_i(\theta+d\theta_-)}{d\theta_-}\right]\frac{1}{d\theta_+ + d\theta_-}\frac{1}{r^2} + \left[\frac{V_i(z+dz_+)}{dz_+} + \frac{V_i(z+dz_-)}{dz_-}\right]\frac{1}{dz_+ + dz_-}\right\} / \left[\left(\frac{1}{\mathrm{d}r_+} + \frac{1}{\mathrm{d}r_-}\right)\left(\frac{1}{\mathrm{d}r_+ + \mathrm{d}r_-}\right) + \left(\frac{1}{d\theta_+} + \frac{1}{d\theta_-}\right)\left(\frac{1}{d\theta_+ + d\theta_-}\frac{1}{r^2}\right) + \left(\frac{1}{dz_+} + \frac{1}{dz_-}\right)\left(\frac{1}{dz_+ + dz_-}\right)\right],$$

where r, θ, z are coordinates, $\mathrm{d}r_+, \mathrm{d}\theta_+, \mathrm{d}z_+$, are step lengths to the next grid points, $\mathrm{d}r_-, \mathrm{d}\theta_-, \mathrm{d}z_-$ step lengths to the previous.

In a 1D spherical coordinate, the potential at a grid point after the ith successive relaxation iteration, V_{i+1}, can be expressed as

$$V_{i+1} = \frac{\rho}{\epsilon}\frac{\mathrm{d}r_+\mathrm{d}r_-}{2} + \frac{V_i(r+\mathrm{d}r_+) - V_i(r-\mathrm{d}r_-)}{r} / \left(\frac{1}{\mathrm{d}r_+} + \frac{1}{\mathrm{d}r_-}\right) + \left[\frac{V_i(r+\mathrm{d}r_+)}{\mathrm{d}r_+} + \frac{V_i(r-\mathrm{d}r_-)}{\mathrm{d}r_-}\right] / \left(\frac{1}{\mathrm{d}r_+} + \frac{1}{\mathrm{d}r_-}\right),$$

where r is the coordinate, $\mathrm{d}r_+$ is the distance to the next grid point, $\mathrm{d}r_-$ the distance to the previous.

In 3D Spherical Coordinate, the potential at a grid point after the ith successive relaxation iteration, V_{i+1}, can be expressed as

$$V_{i+1} = \left\{ \frac{\rho}{2\epsilon} + \frac{2}{r}\frac{V_i(r+\mathrm{d}r_+) - V_i(r+\mathrm{d}r_-)}{\mathrm{d}r_+ + \mathrm{d}r_-} + \frac{1}{r^2\tan\theta}\frac{V_i(\theta+d\theta_+) - V_i(\theta+d\theta_-)}{d\theta_+ + d\theta_-} + \left[\frac{V_i(r+\mathrm{d}r_+)}{\mathrm{d}r_+} + \frac{V_i(r+\mathrm{d}r_-)}{\mathrm{d}r_-}\right]\frac{1}{\mathrm{d}r_+ + \mathrm{d}r_-} + \left[\frac{V_i(\theta+d\theta_+)}{d\theta_+} + \frac{V_i(\theta+d\theta_-)}{d\theta_-}\right]\frac{1}{d\theta_+ + d\theta_-}\frac{1}{r^2} + \left[\frac{V_i(\phi+d\phi_+)}{d\phi_+} + \frac{V_i(\phi+d\phi_-)}{d\phi_-}\right]\frac{1}{dz_+ + dz_-}\frac{1}{r^2\sin^2\theta}\right\} / \left[\left(\frac{1}{\mathrm{d}r_+} + \frac{1}{\mathrm{d}r_-}\right)\left(\frac{1}{\mathrm{d}r_+ + \mathrm{d}r_-}\right) + \left(\frac{1}{d\theta_+} + \frac{1}{d\theta_-}\right)\left(\frac{1}{d\theta_+ + d\theta_-}\frac{1}{r^2}\right) + \left(\frac{1}{d\phi_+} + \frac{1}{d\phi_-}\right)\left(\frac{1}{d\phi_+ + d\phi_-}\right)\frac{1}{r^2\sin^2\theta}\right],$$

where r, θ, ϕ are coordinates, $\mathrm{d}r_+, \mathrm{d}\theta_+, \mathrm{d}\phi_+$, are the step lengths to the next grid points, $\mathrm{d}r_-, \mathrm{d}\theta_-, \mathrm{d}\phi_-$ to the previous.

References

1. D. Radford, icpc_siggen. https://github.com/radforddc/icpc_siggen
2. B. Bruyneel, B. Birkenbach, P. Reiter, Eur. Phys. J. A **52**, 3 (2016)
3. M. Salathe, Study on modified point contact germanium detectors for low background applications. Ph.D. thesis, Ruprecht-Karls-Universität Heidelberg (2015)
4. G.K. Giovanetti, P-type point contact germanium detectors and their application in rare-event searches. Ph.D. thesis, University of North Carolina (2015)
5. I. Abt, A. Caldwell, D. Lenz, J. Liu, X. Liu, B. Majorovits, Eur. Phys. J. C **68**, 609 (2010)
6. D. Radford, Large inverted-coaxial point-contact germanium detectors (2017). https://conferences.lbl.gov/event/121/session/4/contribution/18
7. GERDA Collaboration, M. Agostini et al., Nucl. Part. Phys. Proc. **1876**, 273–275 (2016). 37th International Conference on High Energy Physics (ICHEP)
8. M.J.D. Collaboration, N. Abgrall et al., Adv. High Energy Phys. **2014**, e365432 (2014)
9. CoGeNT Collaboration, C.E. Aalseth et al., Phys. Rev. Lett. **106**(13), 131301 (2011)
10. S. Kerman, V. Sharma, M. Deniz, H. Wong, J.W. Chen, H. Li, S. Lin, C. Liu, Q. Yue, Phys. Rev. D **93**, 113006 (2016)
11. Q. Yue, K. Kang, J. Li, H.T. Wong, J. Phys. Conf. Ser. **718**(4), 042066 (2016)
12. AGATA, https://www.agata.org
13. GRETA, http://greta.lbl.gov
14. D. Radford, Majorana siggen (2015). https://indico.mpp.mpg.de/event/3121/session/6/contribution/33
15. D. Weisshaar et al., Nucl. Instrum. Methods Appl. (2016)
16. SIMION Ion and Electron Optics Simulator, http://simion.com/
17. ANSYS Maxwell: low frequency electromagnetic fields. http://www.ansys.com/products/electronics/ansys-maxwell
18. M. Boswell et al., IEEE Trans. Nucl. Sci. **58**(3), 1212 (2011)
19. FEniCS, https://fenicsproject.org/
20. Solid state detector field and charge drift simulation in Julia: JuliaHEP/SolidStateDetectors.jl (2019). https://github.com/JuliaHEP/SolidStateDetectors.jl
21. LEGEND, http://legend-exp.org/
22. Geant4 Collaboration, S. Agostinelli, et al., Nucl. Instrum. Methods A **506**(3), 250 (2003)
23. Geant4 Collaboration, J. Allison et al., IEEE Trans. Nucl. Sci. **53**(1), 270 (2006)
24. J. Allison et al., Nucl. Instrum. Methods A **835**, 186 (2016)
25. J. Liu, Development of segmented germanium detectors for neutrinoless double beta decay experiments. Ph.D. thesis, Technische Universität München (2009). https://mediatum.ub.tum.de/node?id=701884
26. D. Lenz, Pulse shapes and surface effects in segmented germanium detectors. Ph.D. thesis, Technische Universität München (2010). https://inis.iaea.org/search/search.aspx?orig_q=RN:41120150
27. I. Abt, A. Caldwell, K. Kroeinger, J. Liu, X. Liu, B. Majorovits, Nucl. Instrum. Methods A **583**(2–3), 332 (2007). https://doi.org/10.1016/j.nima.2007.09.017
28. P. Luke, F. Goulding, N. Madden, R. Pehl, I.E.E.E. Trans, Nucl. Sci. **36**(1), 926 (1989)

29. Julia Language. https://julialang.org/
30. W.L. Hansen, E.E. Haller, MRS online proceedings library archive **16** (1982)
31. G. Wang, M. Amman, H. Mei, D. Mei, K. Irmscher, Y. Guan, G. Yang, Mater. Sci. Semiconduct. Process. **39**, 54 (2015)
32. Conjugate gradient method—wikipedia. https://en.wikipedia.org/wiki/Conjugate_gradient_method
33. ROOT a data analysis framework. https://root.cern.ch/
34. M. Salathe, R.J. Cooper, H.L. Crawford, D.C. Radford, J.M. Allmond, C.M. Campbell, R.M. Clark, M. Cromaz, P. Fallon, P.A. Hausladen, M.D. Jones, A.O. Macchiavelli, J.P. Wright, Nucl. Instrum. Methods A **868**, 19 (2017). https://doi.org/10.1016/j.nima.2017.06.036
35. L. Mihailescu, W. Gast, R. Lieder, H. Brands, H. Jäger, Nucl. Instrum. Methods A **447**(3), 350 (2000). https://doi.org/10.1016/S0168-9002(99)01286-3
36. L. Reggiani, Phys. Rev. B **17**(6), 2800 (1978). https://doi.org/10.1103/PhysRevB.17.2800
37. B. Bruyneel, P. Reiter, G. Pascovici, Nucl. Instrum. Methods A **569**(3), 764 (2006)
38. Z. He, Nucl. Instrum. Methods A **463**(1), 250 (2001). https://doi.org/10.1016/S0168-9002(01)00223-6
39. V. Radeka, Ann. Rev. Nucl. Part. Sci. **38**(1), 217 (1988). https://doi.org/10.1146/annurev.ns.38.120188.001245
40. E. Gatti, G. Padovini, V. Radeka, Nucl. Instrum. Methods Phys. Res. **193**(3), 651 (1982). https://doi.org/10.1016/0029-554X(82)90265-8
41. T. Nashashibi, G. White, I.E.E.E. Trans, Nucl. Sci. **37**(2), 452 (1990). https://doi.org/10.1109/23.106661
42. T. Nashashibi, Nucl. Instrum. Methods A **322**(3), 551 (1992). https://doi.org/10.1016/0168-9002(92)91230-7
43. ROOT coding conventions. https://root.cern.ch/coding-conventions
44. Y. Takahashi, V. Vassilev, O. Shadura, R. Isemann, EPJ Web Conf. **214**, 02011 (2019). https://doi.org/10.1051/epjconf/201921402011
45. GitHub. https://github.com/
46. The MIT license | open source initiative. https://opensource.org/licenses/MIT
47. GitHub flavored markdown spec. https://github.github.com/gfm/
48. Doxygen. http://www.stack.nl/~dimitri/doxygen/index.html
49. Cling. https://cdn.rawgit.com/root-project/cling/master/www/index.html
50. Project Jupyter. https://www.jupyter.org
51. R. Barrett, et al., in *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd edn*. (SIAM, Philadelphia, 1994)
52. M. Amman, (2018). arXiv:1809.03046