Efficient computation paths for the systematic analysis of sensitivities

Authors: Paolo Greppi[a*], Elisabetta Arato[b]

(a) libpf.com, Via Cesati 12 I-13100 Vercelli Italy, paolo.greppi@libpf.com, telephone +39 320 8960642
(b) University of Genoa, Department of Civil, Environmental and Architectural Engineering, Via Opera Pia, 15 I-16145 Genova Italy, elisabetta.arato@dicat.unige.it
(*) Corresponding author

Abstract

A systematic sensitivity analysis requires computing the model on all points of a multi-dimensional grid covering the domain of interest, defined by the ranges of variability of the inputs.
The issues to efficiently perform such analyses on algebraic models are handling solution failures within and close to the feasible region and minimizing the total iteration count. Scanning the domain in the obvious order is sub-optimal in terms of total iterations and is likely to cause many solution failures.
The problem of choosing a better order can be translated geometrically into finding Hamiltonian paths on certain grid graphs.
This work proposes two paths, one based on a mixed-radix Gray code and the other, a quasi-spiral path, produced by a novel heuristic algorithm. Some simple, easy-to-visualize examples are presented, followed by performance results for the quasi-spiral algorithm and the practical application of the different paths in a process simulation tool.

Keywords

Process simulation; sampling; sensitivity analysis; convergence; computation paths; multi-dimensional grids.

1. Introduction

One of the best ways to understand the capabilities and limitations of a model of a physical, chemical or social system is to study the sensitivities to the inputs.

But the term sensitivity analysis is used with different meanings in physical sciences, engineering, econometrics and statistics. The differential or local sensitivity [1, 2] involves the calculation of derivatives of the model, which can be also used for numerical solution and optimization. The

global sensitivity for uncertainty analysis [3 – 7] is based on random or pseudo-random sampling of the inputs of the model. A third meaning that is more common in engineering disciplines [8 – 10] involves systematically computing the value of the model on all points of a multi-dimensional grid that covers the domain of interest; this type of sensitivity analysis can be called a "systematic sensitivity analysis". Indeed systematic sensitivity analyses have a variety of applications:

- To find out what the feasible ranges are for solving the model;
- To study the non-linear sensitivity of the outputs on the inputs;
- To generate performance maps or look-up tables that can be interpolated with simpler, explicit algebraic functions or supplied to neural networks for training;
- To identify regions with different modes of operation;
- To visualize the model predictions graphically using 3-D or parametric plots;
- To test the model or the model performance (execution time) etc.

When the model is formulated algebraically, from the mathematical point of view it is a complex, multi-variable function; often the function is implicit, and the resulting large-scale system of non-linear algebraic equations (NLAE) has to be solved iteratively. Each function evaluation is therefore expensive in terms of processing time, and the required number of iterations is dependent on how close the next point is to the last solved point, as the current results are used as starting values for the unknowns.

If the starting point is too distant from the desired solution, the NLAE solver may fail; for resiliency, after a failure the solver state must be recovered by resetting or reverting to a previous successful solution - but that comes at a cost. Finally, a solution may exist only in a subset of the domain (the feasible region), whose form and extension is rarely known *a priori*. The domain of interest where the systematic sensitivity analysis is computed typically overlaps and encloses the feasible region but also inevitably includes some non-feasible points. The solver will almost certainly fail at non-feasible points, and it is also likely to fail at points close to the unfeasible region - but often that is precisely where the interesting results lie.

The most obvious way to perform a systematic sensitivity analysis is to scan the multi-dimensional grid one "line" at a time, in the same way we scan the lines of text when we read. On a two-dimensional grid, the resulting path looks like a zigzag line (see Figure 1).

The zigzag path can easily be generalized to higher dimensions as shown below, but it is sub-optimal in two aspects:

1) The "carriage returns" when a line scan is complete and the next line is started cause discontinuities ("jumps") with a large change to one or more manipulated variables, which

go from their maximum values back to the minimum;

2) In general before the start of the sensitivity, the solver state may be such that the current values for the manipulated variables lie somewhere inside the domain, so there is one additional discontinuity (visualized in Figure 1 by the starting arc) to move to the first point of the sensitivity located on the boundary of the domain;
3) The central region of the domain will likely be feasible, but the regions close to the border could be unfeasible; the zigzag path will visit the borders very soon, potentially causing early NLAE solver failures.

So it would be convenient to find different, better orders for visiting the points on the grid and performing the systematic sensitivity analysis while minimizing discontinuities and visiting the borders as late as possible.

2. Proposed method

To reduce the discontinuities the first improvement is to switch from "left-to-right" or "right-to-left" reading to boustrophedon reading. In boustrophedon reading lines are alternatively read "left-to-right" and "right-to-left" (see Figure 2). In this way only one manipulated variable is changed at a time, and the variable changes are minimal with the exception of the initial discontinuity (second issue above).

It turns out that there is a large class of paths that fulfill the two requirements of changing only one manipulated variable at a time, and keeping the variable changes minimal. To describe this class of paths, it is necessary to introduce more detailed algebraic and geometrical descriptions, with their notation and definitions.

Assuming that *n* input variables of the steady-state process model are manipulated, variable 0 is varied over m_0 points identified with integers running between 0 and m_0-1, variable 1 over m_1 points between 0 and m_1-1, variable 2 over m_2 points between 0 and m_2-1 ... and finally variable *n*-1 is varied over m_{n-1} points between 0 and $m_{n-1}-1$. In the algebraic description each of the $\prod_{i\in[0..n-1]} m_i$ points in the resulting multi-dimensional grid can be identified by a vector of n positive integers (a "tuple" i.e. an ordered list of elements), each with a different maximum constraint. This algebraic representation also highlights the fact that the grid is a subset of the $\mathbb{Z}^n$ lattice, the space of vectors of positive integers.

The a_i elements of this vector can be written as row-vectors in the straightforward order of an increasing index $(a_{0,}a_{1,}a_2...a_{n-1})$, but for the purpose of this application it is more suitable to write them as row-vectors in order of a decreasing index $(a_{n-1}...a_{2,}a_{1,}a_0)$. This notation is

consistent with the big-endian convention for ordering of individually addressable sub-components within a longer data item in computing, and with the ordering commonly used in writing numbers. This in turn suggests that the tuple written in decreasing index order can be interpreted as a mixed-radix number. A side effect of this interpretation is that an ordering on the points in the grid can be automatically obtained by sorting these numbers lexicographically, i.e. by counting in increasing order. The lexicographical ordering is the same as the zigzag path mentioned above, and provides its generalization for more than two dimensions.

To describe the boustrophedon path rigorously and to generalize it to a more general class of well-behaved paths that avoid discontinuities a geometrical description based on graph theory is required.

"Well-behaved" paths are those that fulfill the requirement of keeping the variable changes at each step minimal. In "strictly well-behaved" paths only one manipulated variable can be changed by one grid unit in each step, i.e. only moves to adjacent points are acceptable and no "diagonal" moves. In "loosely well-behaved" paths one or more manipulated variables can be simultaneously changed by one grid unit, i.e. only moves to adjacent points are allowed, including "diagonal" moves.

The allowed moves can be visualized on the $\mathbb{Z}^n$ lattice by connecting the points that can be reached from each point. If no diagonal moves are allowed and all adjacent points are connected with orthogonal edges, then each internal point has $2 \cdot n$ neighbors (points on the boundary have $2 \cdot n - 1 \ldots n$ neighbors depending on the dimension of the boundary) and the resulting undirected graph is called grid graph, see [11]. A grid graph in two-dimensions is shown in Figure 3. If it were permitted to change a manipulated variable from the last value back to the first value and *vice versa* (what amounts to a very large discontinuity), then all points would have $2 \cdot n$ neighbors, and the grid graph would be embedded on a multi-dimensional torus rather than on the Euclidean space.

If diagonal moves are allowed and the non-orthogonal edges between adjacent points are added to the grid graph, then each internal point has $3^n - 1$ neighbors (points on the boundary may have $2 \cdot 3^{n-1} - 1 \ldots 2^n - 1$ neighbors depending on the dimension of the boundary) and what can be defined as a “dense grid graph” is obtained. A dense grid graph in two-dimensions is shown in Figure 4.

On this basis it is possible to rigorously define a strictly well-behaved path as a Hamiltonian path (a path that visits each vertex exactly once) on the grid graph, and a loosely well-behaved path as a Hamiltonian path on the dense grid graph. In the literature on databases and image compression, several Hamiltonian paths on the grid graph are described, and sometimes called space-filling curves: such as the Peano curve, the Hilbert curve and the Z-curve, see for example [12]. But these curves are not relevant here because their properties are not helpful for this application.

Returning to an algebraic point of view, Hamiltonian paths on grid graphs are also related to mixed-

radix Gray codes [13]. A Gray code [14] is an ordering of integers where two successive values differ by only one digit. The original Gray code was proposed in [15] to solve certain communication problems for binary digits. When Gray codes are applied to digits in bases higher-than-two, there are two possibilities: non-modular Gray where the transitions 0 to m-1 and m-1 to 0 are not acceptable, and modular Gray where these transitions are acceptable; this distinction does not make sense for binary digits since 0 to 1 and 1 to 0 are the only two possible transitions. Modular Gray codes correspond to Hamiltonian paths on grid graphs embedded on tori so they should be ruled out due to their large discontinuities; on the other hand, mixed-radix, or more precisely $(m_{0,}m_{1,}m_2...m_{n-1})$ -radix non-modular Gray codes match the definition of strictly well-behaved paths above. There are many possible binary Gray codes, and even more possible $(m_{0,}m_{1,}m_2...m_{n-1})$ -radix non-modular Gray codes. The simplest one is the reflected mixed-radix Gray code, where each digit moves from 0 to m_i and back to 0 and so on, and the least-significant digits are changed more often.

A loop-less algorithm to generate this sequence and conversions with ordinary numbers are provided in [16]. An algorithm to generate the sequence with loops is presented in [17]. The reflected mixed-radix Gray code ordering is coincident with the boustrophedon path proposed above, and provides its generalization for more than two dimensions.

Mixed-radix non-modular Gray codes solve the first issue mentioned above in paragraph 1. But the second issue of avoiding the discontinuity on start, and the third issue of delaying visiting the borders for as long as possible, are still to be addressed. Both the lexicographical ordering and the reflected mixed-radix Gray code ordering visit the borders of the domain too early and too often: they actually start from the boundary. What is required to address both issues is a Hamiltonian path on the dense grid graph that starts near the center of the domain and moves spiraling outwards, so that the most extreme regions are visited as late as possible. Such a path is trivial to draw on the two-dimensional grid as a square spiral (Figure 5). But the square spiral has the limitation that it can only cover grids, completely and without overflows, if the number of points in one direction differs by zero or one from the number of points in the other direction. Also there is no known generalization of the square spiral to higher dimensions.

For these reasons a novel heuristic algorithm was developed that enumerates the points on the grid in a sequence that approximates the multi-dimensional spiral with a quasi-spiral. It is based on a definition for the distance between points in the sense of the infinity-norm $\|.\|_\infty$, i.e. the positive integer obtained as the maximum absolute value of the difference of any digit: $\max_i |a_i - b_i|$.

The quasi-spiral path tries to monotonically increase the distance from the initial point while the distance of each point from the previous one (if possible) equals 1: the relaxation of the requirement

that it should be a Hamiltonian path on the dense grid graph is necessary to avoid dead-locks. The algorithm is simple and only relies on integer arithmetic; it allows the pre-computation of the direct and reverse mapping between the lexicographical ordering and the quasi-spiral. Once this initialization has been performed, increment and look-up can be performed. The algorithm pseudo-code is reproduced in the Appendix.

3. Algorithm implementation

The algorithm has been implemented in C++ and integrated in LIBPF™ 1.0 (LIBrary for Process Flowsheeting), [18], a process flow-sheeting and modeling tool arranged as a C++ library.

The typical architecture of a LIBPF™ application for interactive process flow-sheeting (see Figure 6) consists of two processes, one user interface process and one calculation kernel; both processes interact with a relational database for data persistency and transfer. The user interface will start a kernel process when a calculation is requested by the user, sending certain basic information as arguments on the command line. The two processes interact via the database for exchanging inputs, configuration settings and results, so the direct inter-process communication is limited to controlling the second process (starting and stopping), handling the process start and finish events and processing the diagnostic output.

To implement the multi-dimensional sensitivity analysis, the direct inter-process communication between the user interface and the calculation kernel was extended by providing a supplementary data transfer mechanism, based on two files encoded as Extensible Markup Language (XML) according to [19]. The first XML file is for sending the sensitivity input from the user interface to the calculation kernel, and the second XML file is for sending the sensitivity results from the calculation kernel back to the user interface. For documentation and for enforcing constraints on their structure and content both files have an associated XML Schema, in accord with [20].

To maximize interactivity, results are fed back to the user interface as soon as they are available, by having the kernel generate a new sensitivity results XML file after each steady-state process model evaluation, and by having the user interface react to a file-changed event emitted by the file-system to rescan the XML file and load the new results; furthermore the sensitivity analysis can be stopped and restarted and the user can interactively change the sensitivity analysis calculation order between lexicographical order, reflected decimal Gray code and quasi-spiral.

Due to the split architecture of LIBPF™, it was necessary to duplicate most of the code related to the sensitivity ordering on both the user interface and the kernel sides, since the former must be aware of how to convert between the row ordering on the sensitivity results as they are shown to the user (lexicographical order, albeit little-endian) and the calculation order, while the latter must be able to enumerate the points in the calculation order from any point of the sequence (in case of

restarts).

The LIBPF™ User Interface is open source so its source code can be obtained from [21] and reused in accord with [22].

4. Results and discussion

In paragraph 4.1 some simple examples of grids with a few tens of points are presented, to visualize the different paths.

Next, in paragraph 4.2, the performance results for the novel quasi-spiral heuristic algorithm when applied to the enumeration of large sensitivity analysis grids are shown.

Finally, in paragraph 4.3, the three orderings are applied to real-world examples of systematic sensitivity analyses of steady-state process models.

4.1 Simple examples

In the first examples only $n = 3$ input variables are manipulated, resulting in easy-to-visualize three-dimensional grids. Furthermore all the m_i are set equal to 3, so that base-three digits are used to identify the coordinates of each point. This means that each variable is varied over $m = 3$ points between 0 and $m - 1 = 2$, and the $\prod_{i\in[0,1,2]} 3=27$ points in the resulting three-dimensional grid are identified by tuples that can be interpreted as ternary numbers. For example the tuple 123 has a_2 =1, a_1 =2 and a_0 =3, consistent with the big-endian or decreasing index notation.

The lexicographical ordering for the (3,3,3)-grid is listed in Table 1; in these tables the digits that change in the transition from one point to the next are highlighted in bold: note how 1, 2 or even 3 digits are changed at once in the lexicographical ordering. The same path is visualized in Figure 7; in all these figures a color coding for the path is used, with a red-violet-blue-green-yellow-orange sequence used to hint at the order in which the points are visited. The reflected decimal Gray code ordering is listed in Table 2 and is visualized in Figure 8; note how only one digit is ever changed at a time (no diagonals) in Gray code ordering. The quasi-spiral path computed with the proposed algorithm starting from the center of the domain is listed in Table 3 and is visualized in Figure 9. Here 1, 2 or even 3 digits are changed at once as well, but in this case the algorithm is successful in avoiding long “jumps”: in fact this path is a Hamiltonian path on the dense grid graph.

Actually the algorithm is not always successful in this sense, as the last example in Figure 10 shows: here the relaxation of the requirement that the path should be a Hamiltonian path on the dense grid graph is exploited to avoid the dead-lock in 40, causing a 4-step jump from 40 to 44.

4.2 Performance

The performance of the quasi-spiral heuristic algorithm was examined when applied to the enumeration of large systematic sensitivity analyses, on a typical low-end workstation. When there are many points or more than three dimensions it is impossible to check graphically or by inspection that the algorithm is really following a well-behaved path and at the same time spiraling outwards, so the criteria on the distance of each point from the previous point and the monotonically increasing distance from the initial point were numerically tested.

The tests with six-dimensional sensitivities (Table 4) having an increasing number of steps in each dimension, show that the steps longer than one (the "jumps", that indicate a violation of the requirement that the path should be a Hamiltonian path on the dense grid graph) are in the order of 1% of the points, decreasing down to 0.2% with three million points. The backward steps, that indicate the spiraling is sometimes inwards rather than outward, happen about 0.5% of the time; sensitivities with an even number of points in each direction seem to cause comparatively more backward steps. When the number of dimensions is constant, there is a linear dependence of the execution time on the number of points.

When the same number of points (about one million) is enumerated by changing the number of dimensions (Table 5) the frequency of "jumps" is constant at about 0.2%, while the frequency of backward steps decreases linearly with the dimensions. This observation can be explained by the increased number of escapes and shortcuts available in higher-dimensional spaces.

When the number of points is constant, the execution time displays an exponential dependence on the number of dimensions.

The algorithm execution time is less than one second for a practical number of points (less than 10000).

4.3 Practical examples

The performance of the different paths has been tested against two real-world examples with LIBPF™.

The first practical example is the flash of a mixture of four linear-chain hydrocarbons (C_3 to C_6) over a wide range of compositions, at fixed vapor fraction and pressure, using a cubic equation of state. In this case the problem only has a feasible solution at pressures lower than the critical pressure of the mixture, which in turn depends on the composition. The three-dimensional sensitivity analysis was performed with 21 steps in the molar fraction of each of the first three components (a total of 9261 points), varying them between 0.00001 and 0.325388. The results, in terms of performance and number of flash failures at vapor fraction equal to 0.9 and pressure levels between 0.1 and 4.5 MPa, are shown in Table 6.

The reflected Gray and the quasi-spiral orderings are both 30% faster than the lexicographical order in the feasible pressure range (0.4 – 2.5 MPa), but there is no definite advantage in the quasi-spiral. At unfeasible pressures (higher than 3.5 MPa) the number of convergence failures increases dramatically, with the reflected Gray ordering performing about one order of magnitude better than the lexicographical order and consistently outperforming the quasi-spiral ordering in terms of total computation time; note that here the performance results are non-reproducible and heavily dependent on the details of the NLAE solver resiliency and error recovery functionality.

For this example all benefits can be reaped with the reflected Gray ordering, and the quasi-spiral ordering has no advantage; the sorting of the components has been chosen so that the starting point for the lexicographical and the reflected Gray paths will be at low molar fractions for all the lighter components, with 99.997% of C_6: a composition which very likely has the highest possible mixture critical pressure. Also due to the big-endian convention, the molar fraction of the lightest component (C_3), which is represented by the "most-significant" digit, will be increased last. In other words, the non-spiral orderings will start from the safe side, and carefully creep towards the non-feasible area. The quasi-spiral ordering on the other hand will start around the center of the domain, so in this case it is likely to hit the non-feasible regions first.

The second practical example is based on the hybrid molten carbonate fuel cell-gas turbine (MCFC-GT) energy plant described in [23]. In that reference a two-dimensional sensitivity was performed to study the effect of the operating pressure and MCFC fuel feed flow-rate on the total net electricity output and the plant electrical efficiency. A square spiraling path was used to scan the domain because it turned out to be more robust than other orderings.

One of the main outcomes of that work was that scaling down the electrical power output resulted in a decrease in efficiency, because of the limited flexibility of the gas turbine sub-system (GTS). Consequently it was suggested in the conclusions that the control strategy of the GTS should be improved. This hinted at including the nominal compressor mass flow as a third parameter in the analysis, assuming to control the flow/load of the GTS by manipulating the geometry of the turbo-machinery in addition to the frequency as was the case in the reference.

A new, three-dimensional sensitivity analysis was performed on the same MCFC-GT energy plant, varying not only the pressure and fuel flow but also the nominal compressor mass flow, with 19 steps in each of the three variables (a total of 6859 points). The input parameters and their ranges are shown in Table 7, the results in terms of performance and number of convergence failures in Table 8.

The results show that the reflected Gray and the quasi-spiral orderings outperform the lexicographical one by a factor of two or three in terms of both total calculation time and fraction of

converged points, whereas the quasi-spiral and the reflected Gray orderings behave equivalently.
It should be noted that in both examples the starting point and the extension of the domain have been chosen with a good understanding of the form of the feasible region.

5. Conclusions

Performing a systematic sensitivity analysis of an algebraic, non-linear model by scanning the grid along a lexicographically sorted path is not the best option because of the discontinuities, large changes in manipulated variables and visiting the border regions too early.
Two alternative well-behaved orderings are proposed in this work, one based on the reflected mixed-radix Gray code and the other, a quasi-spiral path produced by a novel heuristic algorithm. The heuristic algorithm is simple and only relies on integer arithmetic to generate the quasi-spiral; its execution time is negligible for practical grids, and has a linear dependence on the number of points for a constant number of dimensions, and an exponential dependence on the number of dimensions for a constant number of points.
The two alternative orderings have been implemented in a process simulation tool, demonstrating significant performance and stability gains with respect to the lexicographical path, when applied to real-world problems. The reflected mixed-radix Gray code is optimal if the shape of the feasible region is known in advance so that the path can start from a safe point. The quasi-spiral is most suited for the initial assessment phases, when the form of the feasible region is unknown, and sensitivity analysis are used to explore its boundaries.

Appendix

The algorithm pseudo-code to generate the quasi-spiral path is:

```
round = 0
current = initial
add current point to the set of visited points
direct[round] = current
reverse[current] = round
increment round
while there is any point left to visit, start
  set the acceptable distance to 1, i.e. start looking for points adjacent to
the current point in the sense of the dense grid
  clear the list of suitable points
  while the list of suitable points is empty, start
    check all neighbors which have distance from the current point lower or
equal to the acceptable distance
```

```
    if the neighbor has not jet been visited
      add the neighbor to the list of suitable points
    increase by 1 the acceptable distance
  end
  sort the list of suitable points in order of increasing distance from the
initial point
  set the current point to the first suitable point
  add current point to the set of visited points
  direct[round] = current
  reverse[current] = round
  increment round
end
last = current
```

References

[1] P. Stangerup, Implementation of sensitivity calculations in a general-purpose simulation program, Comput. Phys. Commun. 117 (1999) 130, doi: 10.1016/S0010-4655(98)00169-6

[2] I. Kioutsioukis, D. Melas, C. Zerefos, I. Ziomas, Efficient sensitivity computations in 3D air quality models, Comput. Phys. Commun. 167 (2005) 23, doi:10.1016/j.cpc.2003.06.001

[3] E. Hofer, Sensitivity analysis in the context of uncertainty analysis for computationally intensive models, Comput. Phys. Commun. 117 (1999) 21, doi: 10.1016/S0010-4655(98)00153-2

[4] P. Wei, Z. Lu, W. Hao, J. Feng, B. Wang, Efficient sampling methods for global reliability sensitivity analysis, Comput. Phys. Commun. In Press, Corrected Proof, doi: 10.1016/j.cpc.2012.03.014

[5] E. Patelli, H.J. Pradlwarter, G.I. Schuëller, Global sensitivity of structural variability by random sampling, Comput. Phys. Commun. 181 (2010) 2072, doi: 10.1016/j.cpc.2010.08.007

[6] F. Campolongo, A. Saltelli, J. Cariboni, From screening to quantitative sensitivity analysis. A unified approach, Comput. Phys. Commun. 182 (2011) 978, doi:10.1016/j.cpc.2010.12.039

[7] S. Tarantola, W. Becker, D. Zeitz, A comparison of two sampling methods for global sensitivity analysis, Comput. Phys. Commun. 183 (2012) 1061, doi:10.1016/j.cpc.2011.12.015

[8] X. Jiang, S. Li, L. Zhang, Sensitivity analysis of gas production from Class I hydrate reservoir by depressurization, Energy 39 (2012) 285, doi: 10.1016/j.energy.2012.01.016

[9] M.O. Daramola, A.J. Burger, A. Giroir-Fendler, Modelling and sensitivity analysis of a nanocomposite MFI-alumina based extractor-type zeolite catalytic membrane reactor for m-Xylene isomerization over Pt-HZSM-5 catalyst, Chemical Engineering Journal 171(2011) 618, doi: 10.1016/j.cej.2011.04.014

[10] S. Kjørstad Fjeldbo, Y. Li, K. Marthinsen, T. Furu, Through-process sensitivity analysis on the

effect of process variables on strength in extruded Al–Mg–Si alloys, Journal of Materials Processing Technology 212 (2012) 171, doi: 10.1016/j.jmatprotec.2011.08.020,

[11] E. W. Weisstein, Grid Graph. From MathWorld – A Wolfram Web Resource. http://mathworld.wolfram.com/GridGraph.html retrieved April 2012

[12] C. Faloutsos, S. Roseman, Fractals for secondary key retrieval, Proceedings of the eighth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems. 1989; doi: 10.1145/73721.73746

[13] B.D. Sharma, R.K. Khanna, On m-ary Gray code, Inform Sciences, 1978;15(1):31-43 doi:10.1016/0020-0255(78)90020-8

[14] R.W. Doran, The Gray Code, Report CDMTCS-304, University of Auckland Centre for Discrete Mathematics and Theoretical Computer Science (CDMTCS), March 2007

[15] F. Gray, Pulse code communication, U.S. Patent 2632058, 1947

[16] D.E. Knuth, Generating All Tuples and Permutations. In: The Art of Computer Programming. Reading: Addison-Wesley; 2005

[17] D-J. Guan, Generalized Gray Codes with Applications, Proc. Natl. Sci. Counc. Repub. Of China (A) 1998;22:841–848 http://nr.stpi.org.tw/ejournal/ProceedingA/v22n6/841-848.pdf

[18] P. Greppi, LIBPF: a library for process flowsheeting in C++, Proceeding of the International Mediterranean Modelling Multiconference, 2006: 435-440

[19] World Wide Web Consortium. Extensible Markup Language (XML) 1.0 (Fifth Edition). 2008; http://www.w3.org/TR/REC-xml/ retrieved April 2012

[20] World Wide Web Consortium. XML Schema Part 1: Structures Second Edition. 2004; http://www.w3.org/TR/xmlschema-1/ retrieved April 2012

[21] P. Greppi, User Interface for Process Flowsheeting. Gitorious. https://gitorious.org/uipf retrieved April 2012

[22] Free Software Foundation, GNU General Public License v2.0. 1991, http://www.gnu.org/licenses/old-licenses/gpl-2.0.txt retrieved April 2012

[23] P. Greppi, B. Bosio, E. Arato, A steady-state simulation tool for MCFC systems suitable for on-line applications. Int. J. Hydrogen Energ. 33 (2008) 6327, doi: 10.1016/j.ijhydene.2008.07.018

Figures

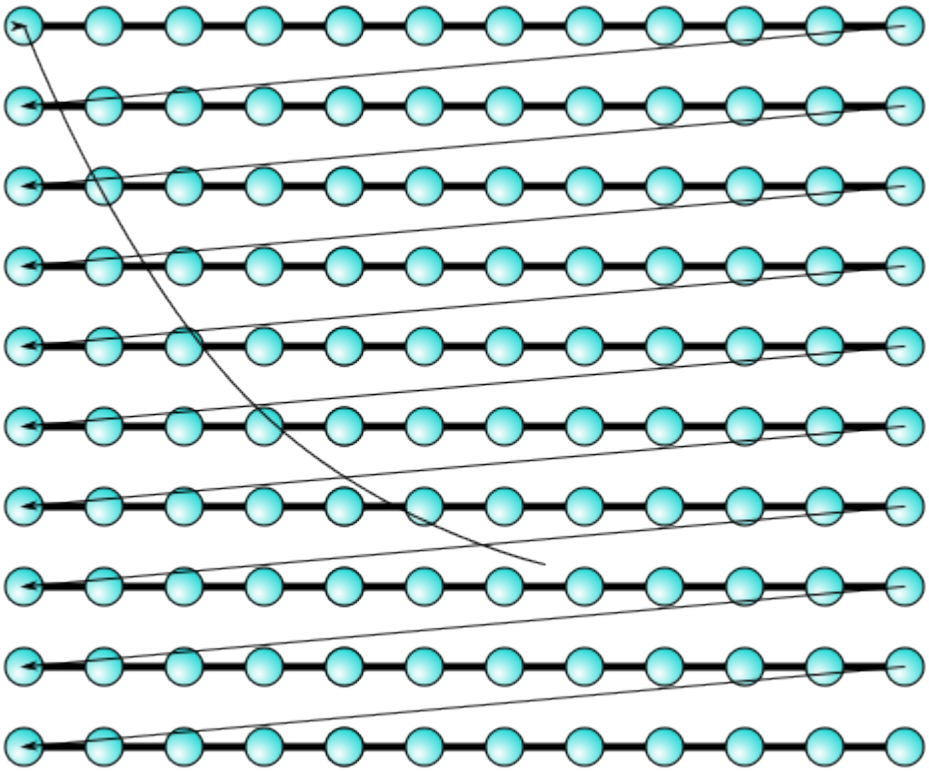

Figure 1 - Zigzag scanning path (lexicographical ordering) in 2 dimensions

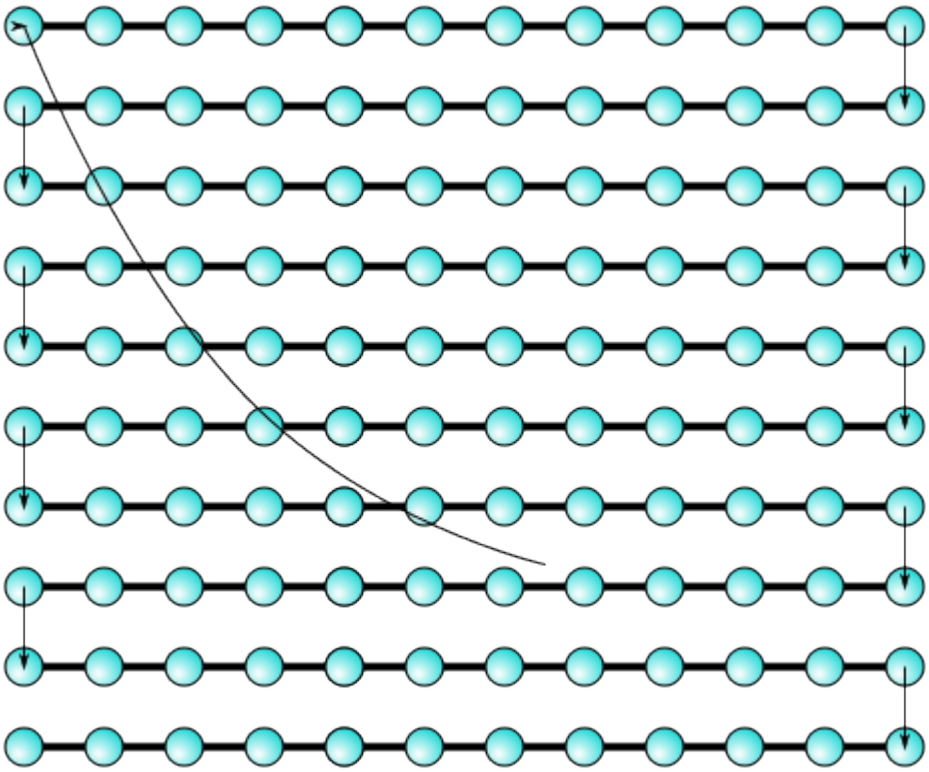

Figure 2 - Boustrophedon scanning path (reflected mixed-radix Gray code ordering) in 2 dimensions

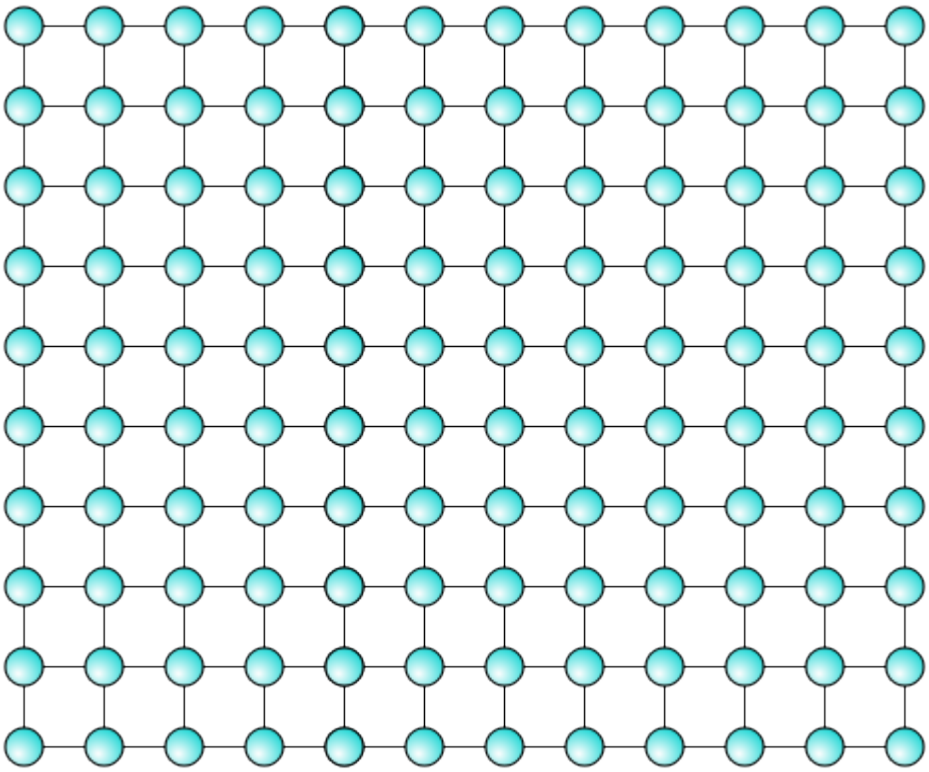

Figure 3 – Grid graph on a small two-dimensional grid

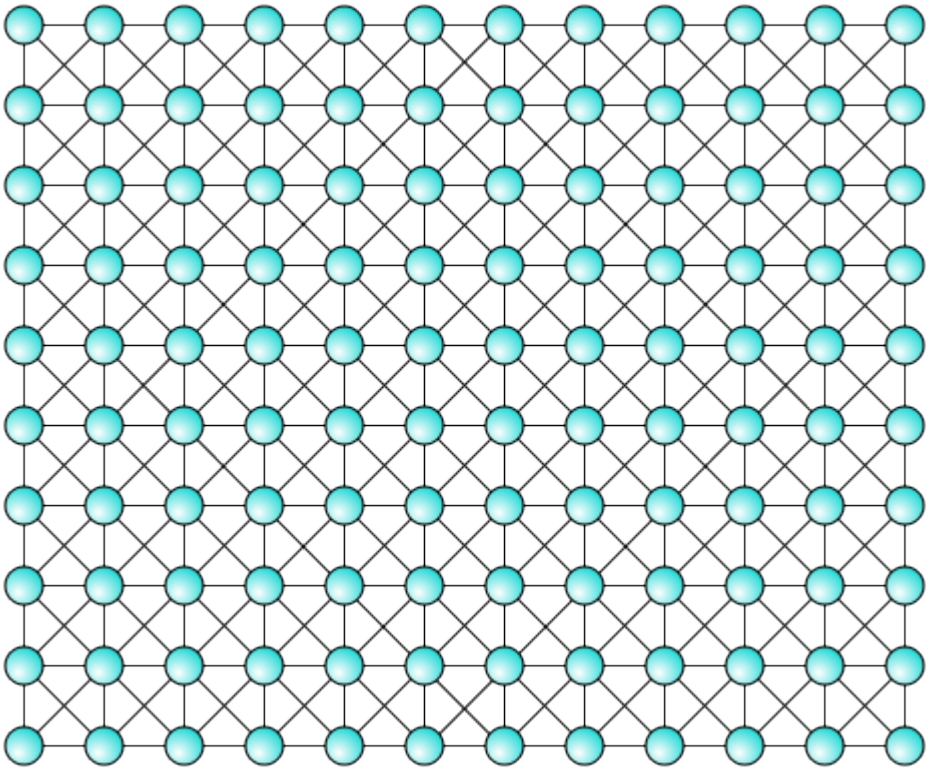

Figure 4 – Dense grid graph on a small two-dimensional grid

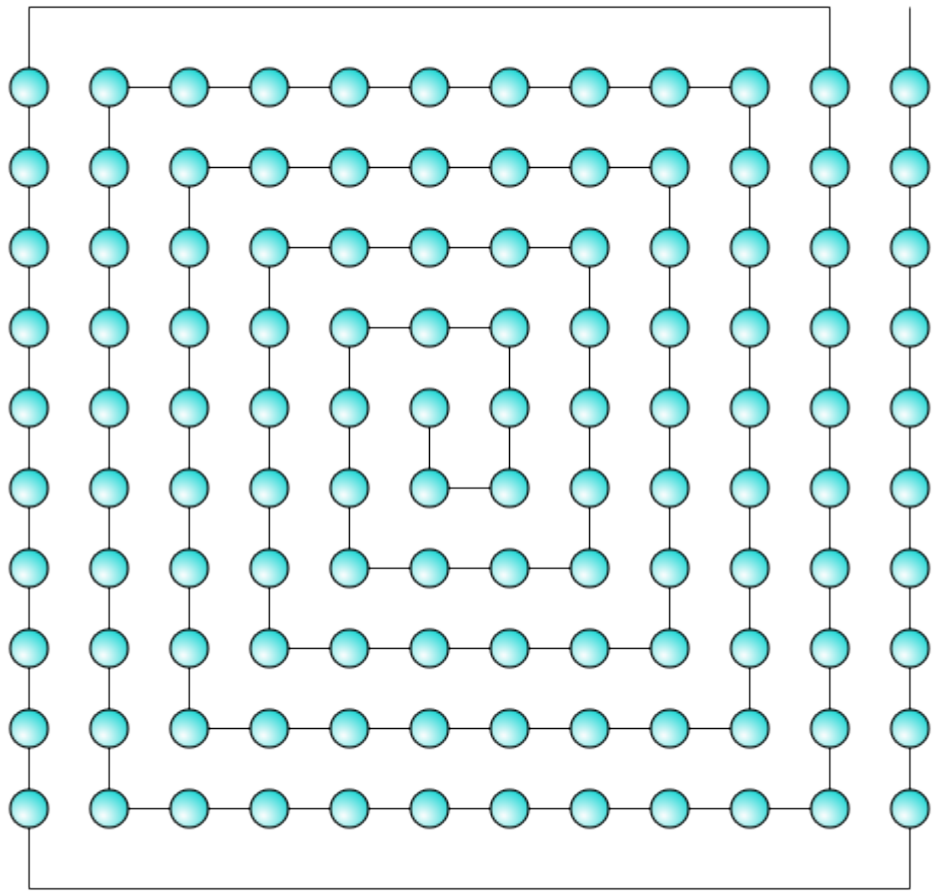

Figure 5 – Square two-dimensional spiral

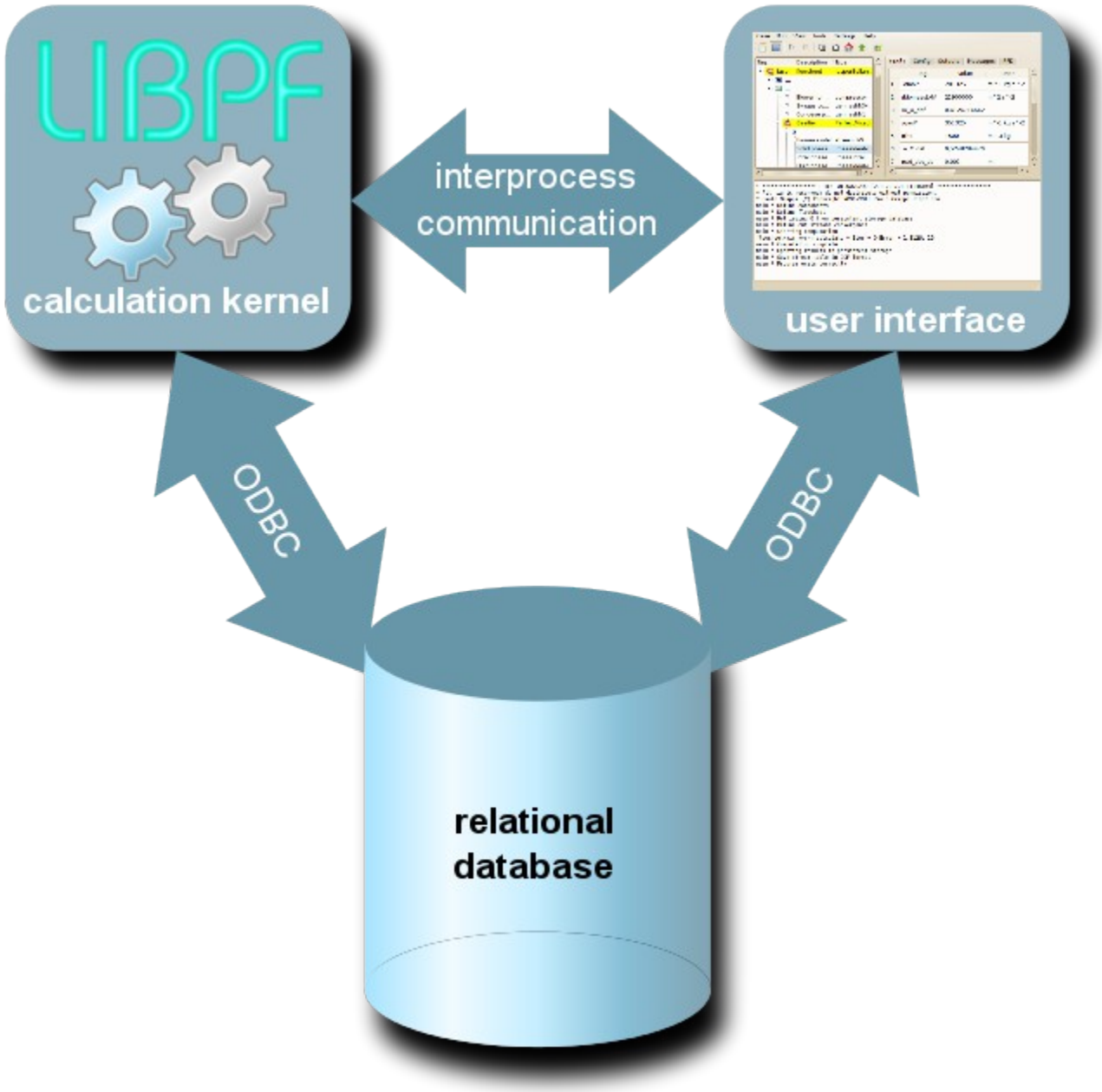

Figure 6 – LIBPF™ standalone architecture

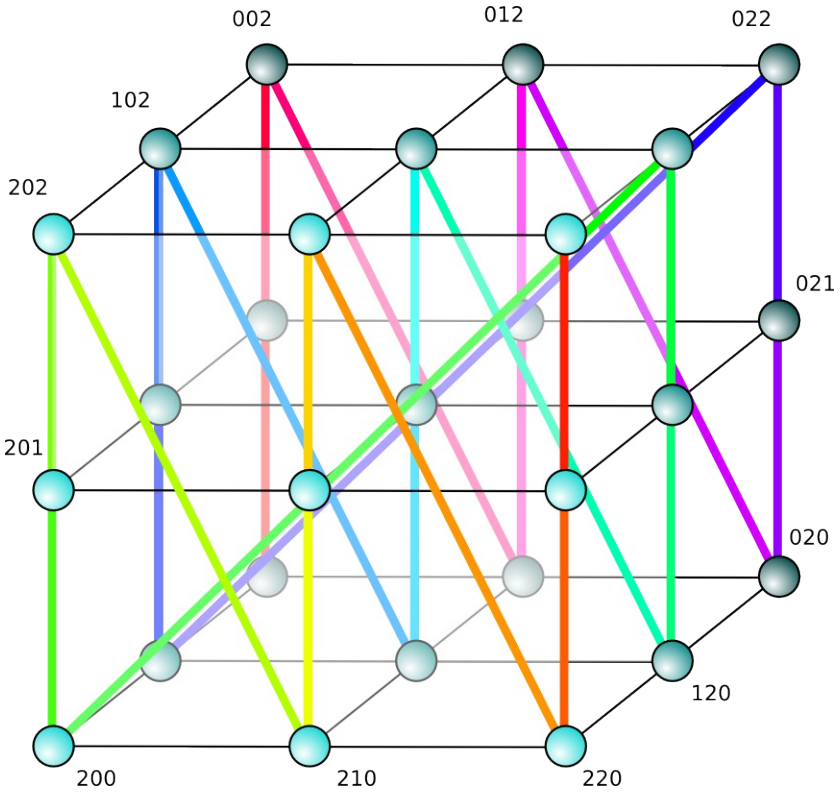

Figure 7 – Lexicographical ordering on a three-dimensional grid having three steps in each direction, with start in point 000 and end in point 222.

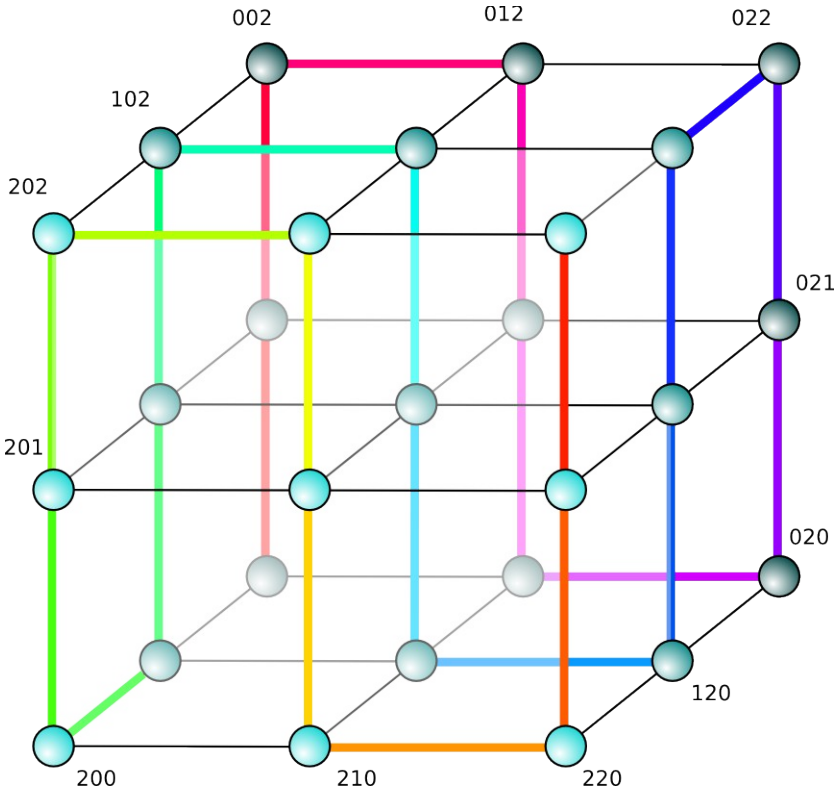

Figure 8 – Reflected Gray code ordering on a three-dimensional grid having three steps in each direction, with start in point 000 and end in point 222.

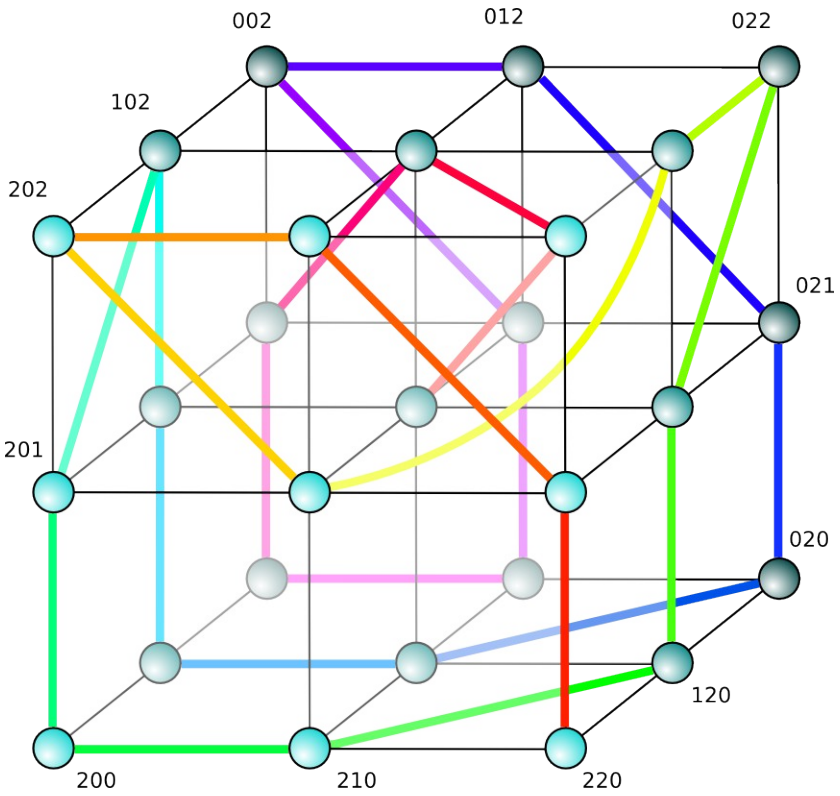

Figure 9 – Quasi-spiral ordering on a three-dimensional grid having three steps in each direction, with start in point 111 and end in point 220.

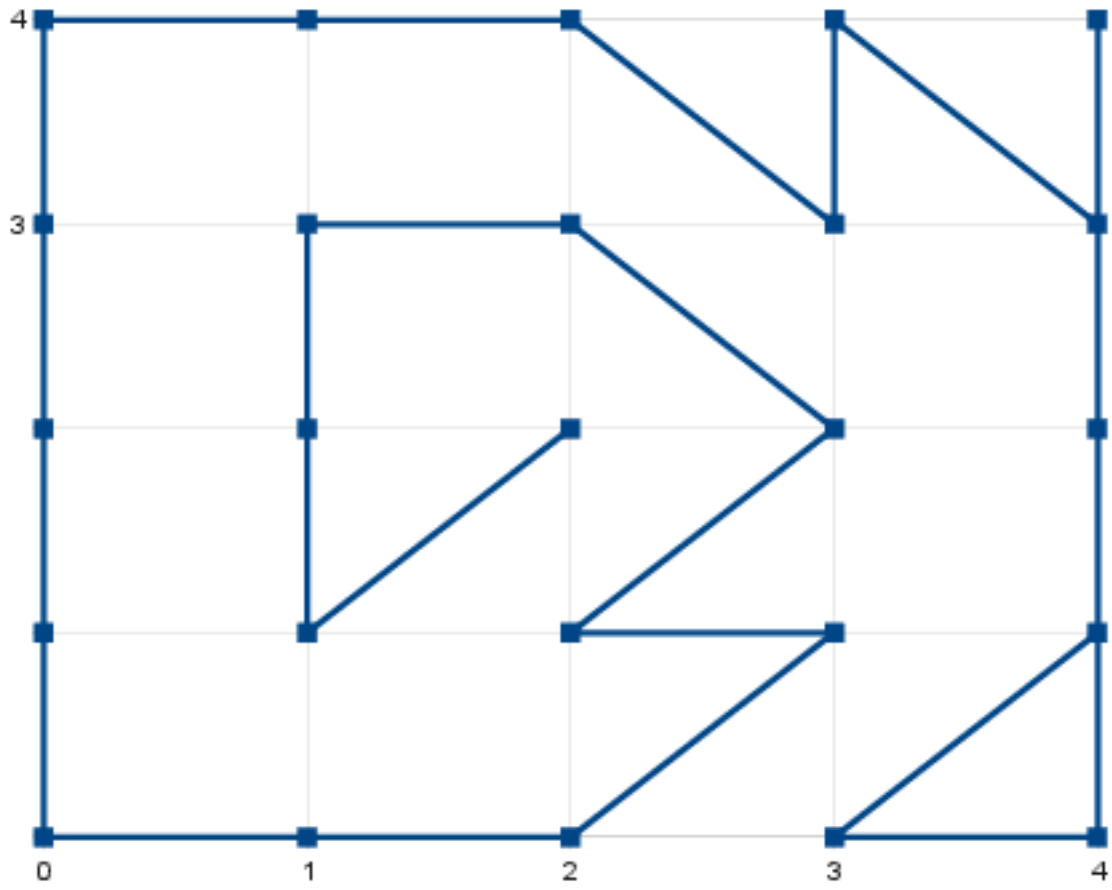

Figure 10 – Quasi-spiral ordering on a two-dimensional grid having five steps in each direction, with start in point 22 and end in points: 43, 42, 41, 30, 40 and 44.

Tables

0	000	9	**100**	18	**200**
1	00**1**	10	10**1**	19	20**1**
2	00**2**	11	10**2**	20	20**2**
3	0**10**	12	1**10**	21	2**10**
4	01**1**	13	11**1**	22	21**1**
5	01**2**	14	11**2**	23	21**2**
6	0**20**	15	1**20**	24	2**20**
7	02**1**	16	12**1**	25	22**1**
8	02**2**	17	12**2**	26	22**2**

Table 1 – Lexicographical ordering for the (3,3,3)-grid.

0	000	9	**1**22	18	**2**00
1	00**1**	10	12**1**	19	20**1**
2	00**2**	11	12**0**	20	20**2**
3	01**2**	12	1**1**0	21	21**2**
4	01**1**	13	11**1**	22	21**1**
5	01**0**	14	11**2**	23	21**0**
6	0**2**0	15	10**2**	24	2**2**0
7	02**1**	16	10**1**	25	22**1**
8	02**2**	17	10**0**	26	22**2**

Table 2 – Reflected Gray code ordering for the (3,3,3)-grid; the digits that change in the transition from one point to the next one are in bold - note how only 1 digit is changed at a time.

0	111	9	0**21**	18	**12**0
1	**222**	10	02**0**	19	12**1**
2	**11**2	11	**11**0	20	**02**2
3	**001**	12	1**0**0	21	**1**22
4	00**0**	13	10**1**	22	**211**
5	0**10**	14	10**2**	23	2**02**
6	01**1**	15	**20**1	24	2**1**2
7	0**02**	16	20**0**	25	2**21**
8	0**1**2	17	2**1**0	26	22**0**

Table 3 – Quasi-spiral ordering for the (3,3,3)-grid, with initial point in the middle of the domain.

Dimensions	**Number of steps in each dimension**	**Total points**	**Steps longer than one (“jumps”)**	**Backward steps**	**Longest step**	**Timing, s**	**Specific timing, s/(1 M points)**
6	1	1	0	0	1	0,008	8000,0
6	2	64	0	0	1	0,009	140,6
6	3	729	15	0	2	0,037	50,8
6	4	4096	31	24	3	0,21	51,0
6	5	15625	131	26	4	1,06	67,7
6	6	46656	228	229	5	3,7	79,3
6	7	117649	600	188	6	10,97	93,2
6	8	262144	830	1266	7	27,9	106,5
6	9	531441	1917	932	8	64,8	121,9
6	10	1000000	2407	4466	9	148,4	148,4

6	11	1771561	5468	3333	10	251,4	141,9
6	12	2985984	5704	13220	11	443,2	148,4

Table 4 – Performance for the enumeration of 6-dimensional sensitivity analysis tables.

Dimensions	**Number of steps in each dimension**	**Total points**	**Steps longer than one ("jumps")**	**Backward steps**	**Longest step**	**Timing, s**	**Specific timing, s/(1 M points)**
3	100	1000000	18	1382	50	16,9	16,9
4	32	1048576	956	8245	22	29,5	28,1
5	16	1048576	2067	9633	15	60,5	57,7
6	10	1000000	2407	4466	9	125,4	125,4
7	7	823543	1980	511	6	230,0	279,2

Table 5 – Performance for the enumeration of sensitivity analysis tables with about one million points and varying number of dimensions.

	Total calculation time, s			**Convergence failures**		
P, MPa	**Lexicographical**	**Reflected Gray**	**Quasi-spiral**	**Lexicographical**	**Reflected Gray**	**Quasi-spiral**
0.1	270	140	183	79	0	27
0.2	225	129	154	36	0	0
0.3	202	161	186	17	0	1
0.4	179	153	151	6	0	0
0.5	169	148	131	4	0	0
1	181	118	123	24	0	0
1.5	200	113	118	33	0	0
2	149	117	113	12	0	0
2.5	146	115	112	16	0	0
3	179	145	118	39	28	3
3.5	836	199	557	468	51	306
4	3992	309	774	2772	91	424
4.5	2455	496	400	1663	215	153

Table 6 – Results for for the three-dimensional sensitivity (9261 points) on the vapor-liquid flash with a cubic equation of state.

Variable	Description	Units	Start	End

P	Operating pressure	MPa	0.4	0.3
S01:Tphase.ndot	Molar flow of natural gas to fuel cell	mol/s	1.67	0.556
mc0	Nominal compressor mass flow	kg/s	1.667	1.111

Table 7 – Input parameters and ranges for the three-dimensional sensitivity on the MCFC-GT energy plant

	Lexicographical	**Reflected Gray**	**Quasi-spiral**
Total calculation time (s)	7447	2586	2850
Convergence failures	73280	1059	21219
Total number of iterations	130746	46402	49372
Fraction of converged points	56 %	98 %	92 %
Average iterations for each converged point	3.8	3.6	4.2

Table 8 – Results for three-dimensional sensitivity (6859 points) on the MCFC-GT energy plant