Virtual Fekete Point Configurations: a case study in perturbing complex systems

Robert van Liere, Jurriaan Mulder, Jason Frank and Jacques de Swart
Center for Mathematics and Computer Science, CWI
Amsterdam, the Netherlands
{robertl,mullie,jason,jacques}@cwi.nl

Abstract

Virtual environments have shown great promise as a research tool in science and engineering. In this paper we study a classical problem in mathematics: that of approximating globally optimal Fekete point configurations. We found a highly interactive virtual environment, combined with a time-critical computation, can provide valuable insight into the symmetry and stability of Fekete point configurations. We believe that virtual environments provide more natural interfaces to complex systems, allowing users to perceive, interpret and interact with the problem more rapidly.

Keywords*: mathematics and visualization, perturbation analysis, explorative visualization, virtual environments*

1. Introduction

An ever increasing demand exists for detailed investigation of complex systems. To simulate real-life phenomena accurately, models are established which can be hard to solve due to their size and complexity. The solutions to these models often involve large data sets, from which it is difficult to extract not only whether the model describes the physical reality accurately, but also whether the model was solved correctly. In order to test the sensitivity of the solution to changes in parameter settings, the amount of data increases even more, because multiple data sets have to be compared. Often it occurs that only a small subset of the data set contains the interesting features of a simulation, but it is difficult to determine this subset beforehand.

Virtual environments have shown great promise for the visualization of these data sets in which exploration plays an important role. Explorative environments allow users, motivated by insufficient knowledge of what is contained in the data, to interactively develop insight into their problem. Due to the interactive nature of these environments, they play a predominant role for gaining insight into complex systems, in which a small perturbation of a parameter setting can result in very different output solutions.

In this paper we study a classical problem in mathematics: that of approximating globally optimal Fekete point configurations. This problem contains many ingredients of a complex system. In the next section we provide some related work. Section 3 provides a mathematical formulation of the Fekete problem, and briefly discusses a very efficient solution method. In section 4 we describe the visualization and interaction techniques used to build the virtual environment. Moreover, we discuss some time-critical performance issues that have been used to meet our performance requirements. Finally, in section 5, we discuss the merits of using virtual environments for analyzing the Fekete problem.

2. Related Work

The analysis and determination of elliptic Fekete point sets has attracted the attention of theoretical and numerical mathematicians and researchers in scientific modeling. The problem was originally proposed by Fekete [1], and has been studied for almost 75 years. It also represents a long-standing numerical challenge: Pardalos states it as one of the open problems in global optimization, [2]. There are many scientific applications that can be modeled as the solution of the Fekete problem (and its possible modifications). Erber and Hockney have studied a problem related to the Fekete problem to find equilibrium configurations of equal charges on a sphere, [3]. Practical applications include problems in structural chemistry, the design of multi-beam laser implosion drives, the optimum placement of communication satellites, and packing and covering problems.

Recently, virtual environments have been used to study mathematical problems and objects. For example, Francis et al. have used distributed CAVEs to study the eversion of a sphere in a collaborative setting, [4]. Roseman has used

the CAVE to display surfaces in 4-dimensional spaces, [5]. Both systems exploit the additional spatial dimension inherent in a virtual environment to get extra information about the structure of the displayed mathematical objects. The focus of our work differs in that we exploit VR interaction styles to gain insight into behavior of complex systems.

Many researchers have used virtual environments for real-time exploration. Bryson [6], for example, has extensively studied the application of virtual reality interfaces in scientific visualization. Although the work has mostly been related to studies in the virtual wind tunnel, the lessons have lead to generalized requirements with regard to implementation issues concerning computation, graphics and data management. Requirements related to real-time performance and natural "anthropomorphic" VR interfaces are discussed in some detail. More recently, experiments have been performed to study time management, time-critical computing and time-critical algorithms, [7, 8].

3. The Fekete Problem

3.1. Formulation

The problem can be formulated as follows: given the unit sphere B in the Euclidean real space $\mathbb{R}^3$, and a positive integer n, find the n-tuple of points (unit length vectors)

$$q(n) = \{q_i,\ i = 1, \ldots, n\}\ , \qquad q_i = (q_{i1}, q_{i2}, q_{i3})^T$$

on the surface S^2 of B, which maximizes the product of distances between all possible pairs $\{q_i, q_j\}$, $1 \leq i < j \leq n$. In other words, we are interested in finding the global maximum of the function

$$f_n(q(n)) = \prod_{1 \leq i < j \leq n} \|q_i - q_j\|\ , \qquad q_i \in S^2$$

where $\|\cdot\|$ indicates the Euclidean norm. A set of vectors $q^*(n) = \{q_i^*,\ i = 1, \ldots, n\}$, where $q_i^* \in S^2$, which satisfies the relations

$$f_n^* = f_n(q^*(n)) = \max_{q(n)} f_n(q(n))\ , \tag{1}$$

is called a set of elliptic Fekete points of order n, [1]. We refer to equation (1) as the Fekete (global optimization) problem.

According to the classical theorem of Weierstrass, the Fekete optimization problem has one or more globally optimal solutions. The traditional approach to computing $q^*(n)$ is with a constrained optimization package. For large n, this can be a very expensive computation, see the comparison in [9].

3.2. Configuration Symmetries

Due to symmetry and rotations, there are infinitely many solution vector sets $q^*(n)$ which satisfy equation (1). In order to compare geometric properties of stable point configurations, it is useful to rotate configurations into a rotation-free canonical form.

For notational purposes we represent arbitrary point configurations in spherical coordinates. The n-tuple $q(n)$—consisting of corresponding unit vectors q_i, $i = 1, \ldots, n$—is denoted by

$$\begin{aligned} q_{i1} &= \cos(\theta_i)\sin(\phi_i) \\ q_{i2} &= \sin(\theta_i)\sin(\phi_i) \\ q_{i3} &= \cos(\phi_i) \end{aligned}$$

in which $0 \leq \theta_i < 2\pi$ and $0 \leq \phi_i \leq \pi$.

For each q_i, $i = 1, \ldots, n$, define the associated partial energy

$$e_i = \ln \prod_{j \neq i} \|q_i - q_j\|. \tag{2}$$

We may use this partial energy to give the points a canonical ordering with respect to certain rotations (see [3]). Suppose that the n points are ordered such that their partial energies form a non-decreasing sequence, that is,

$$e_1 \leq e_2 \leq \ldots \leq e_n.$$

Choose a point from the set with the lowest energy and rotate the configuration so that this point is placed at the north pole $\theta = \phi = 0$. Next choose the point from the set of points with the second lowest partial energy which is closest to the north pole and rotate the entire configuration so that this second charge is at zero longitude, $\theta = 0$.

Two point configurations are deemed equivalent if and only if the positions of the points in the rotation-free canonical form are equivalent.

3.3. Alternative Formulation

Shub and Smale [10] refer to numerical difficulty of finding the globally optimal configuration $q^*(n)$, for a given—not too small—n. Difficulties arise due to several reasons: eg, the above mentioned various symmetries of the function f_n, and—more essentially—its inherent multiextremality. The number of local extrema increases drastically with n [3]. Furthermore, the difference in extremal values of $f^*(n)$ is very small, and the energy landscape is shallow near extrema, making any perceivable numerical solution procedure inherently tedious.

An alternative formulation of the problem has been presented in [9]. The n-tuple $q(n)$ is represented as a set of n repellent particles, which move on the unit sphere under

influence of an adhesion force. To describe the dynamic behavior of the particles, the coordinates of the particles are parameterized by a time variable t. Introducing the velocity vectors p_i, the resulting system can be described by the following set of differential-algebraic equations:

$$\begin{aligned} \dot{q}_i &= p_i \\ \dot{p}_i &= -\kappa p_i - \nabla_{q_i} V(q) - 2q_i\lambda_i \\ 0 &= q_i \cdot q_i - 1 \end{aligned} \tag{3}$$

Here, the notation $\dot{q} \equiv dq/dt$ is used to denote differentiation with respect to time, $-\kappa p$ represents the adhesion force, $2q_i\lambda_i$ represents the normal force associated with the the constraint $\|q_i(t)\| = 1$, and the potential energy of the repellent force field $V(q)$ is defined by

$$V(q) = -\ln \prod_{1\le i<j\le n} \|q_i - q_j\|.$$

Due to the adhesion force, the solution of the dynamical system (3) will approach a steady state such that $\nabla V(q^*) = 0$. If, in addition, $V''(q^*)$ is positive definite, then q^* will be a local minimum of $V(q)$. Clearly, minimization of $V(q)$ is equivalent to maximization of $f_n(q(n))$.

3.4. Solver

A very efficient algorithm is proposed that exploits the special structure and retains certain physical properties of the dynamical system (see [11] for details). To give a flavor of the numerics involved, we briefly discuss this solution method.

The dynamical system (3) can be seen as a constant energy (Hamiltonian) system to which damping has been added. The constrained Hamiltonian dynamics are described by

$$\begin{aligned} \dot{q}_i &= p_i \\ \dot{p}_i &= -\nabla_{q_i} V(q) - 2q_i\lambda_i \\ 0 &= q_i \cdot q_i - 1 \end{aligned} \tag{4}$$

where λ_i is a Lagrange multiplier. Research conducted over the last ten years has shown the importance of preserving the symplectic structure to effectively simulate such a system, [12]. The remaining part of (3) is a system of pure damping:

$$\begin{aligned} \dot{q} &= 0 \\ \dot{p} &= -\kappa p \end{aligned} \tag{5}$$

with exact solution $q(t) = q(0)$, $p(t) = \mathrm{e}^{-\kappa t}p(0)$. Our approach is to use a symmetric splitting method to propagate the solution to (3), with three steps:

1. Solve (5) exactly over an interval of length $h/2$.
2. Solve (4) approximately over an interval of length h with the symplectic leapfrog method described below.
3. Solve (5) exactly over an interval of length $h/2$.

Here it is assumed that the output of each step serves as the input to the succeeding step. Repeated application of the algorithm produces a discrete representation of the solution which is second order accurate in the time stepsize h.

The leapfrog method for the constrained system (4) is defined by first solving simultaneously (for the unknowns q_i^{n+1}, $p_i^{n+1/2}$ and λ_i):

$$\begin{aligned} q_i^{n+1} &= q_i^n + hp_i^{n+1/2} \\ p_i^{n+1/2} &= p_i^n - \tfrac{h}{2}\nabla_{q_i} V(q^n) - 2q_i^n\lambda_i \\ 0 &= q_i^{n+1} \cdot q_i^{n+1} - 1 \end{aligned}$$

for $i = 1, \ldots n$, and then solving simultaneously (for the unknowns $p_i^{n+1/2}$ and μ_i):

$$\begin{aligned} p_i^{n+1} &= p_i^{n+1/2} - \tfrac{h}{2}\nabla_{q_i} V(q^{n+1}) - 2q_i^n\mu_i \\ 0 &= q_i^{n+1} \cdot p_i^{n+1} \end{aligned}$$

for $i = 1, \ldots n$, where the Lagrange multipliers λ_i and μ_i are chosen (in the case of λ_i by solving a scalar quadratic equation) to satisfy the algebraic constraints.

This method is shown to exactly preserve the evolution of the symplectic structure, suggesting that we obtain the "proper" equilibrium configuration associated with a given initial condition.

If the trajectories of two points should make a close approach (or if during interaction two points should be placed nearby each other), the resulting accelerations may cause the algorithm to break down unless the stepsize is adjusted accordingly. To do this while maintaining the symmetric structure of the overall integration scheme requires care. For this purpose we have used an adaptive version of leapfrog based on introducing a transformation of time, [13].

4. Virtual Fekete Point Configurations

We have implemented the Fekete problem in a virtual environment. The implementation consists of two tightly coupled components: the user interface and the computation. Interaction with the computation is performed by changing parameters or the state of a configuration. These changes will immediately be taken into account by the computation; i.e., computational steering is supported.

The goal of the interface is to provide a highly interactive environment which will allow the user to study the dynamics of Fekete point configurations. Since there are multiple rotation-free stable configurations, the user must have the ability to study the sensitivity to perturbation of stable configurations. In this section we present various interactive visualization techniques used to implement the interface. In

section 5, we discuss the merits of using virtual environments for analyzing the dynamics of Fekete point configurations.

4.1. Presentation

Figure 1 shows the basic user interface of the virtual environment. A point configuration is presented as a set of small spheres, a convex hull and trajectories from each sphere. The positions of the spheres indicate the Fekete points. The convex hull is used to visualize connectivity relationships between points. In addition, the convex hull is used as an important depth cue to determine the position of a point. Trajectories show the path the points have taken during a finite number of time steps. The mean velocity of a point can be estimated by examining the length of its trajectory.

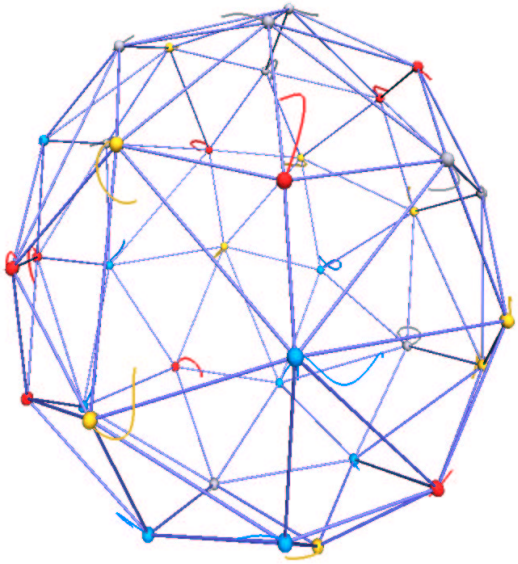

Figure 1. A 36 point configuration with convex hull and trajectories.

The user interface also supports the concept of a *local moonscape*, see figure 2. A local moonscape computes and displays the energy of the configuration in the neighborhood of a point. At a stable configuration the energy values on the local moonscape will always be greater or equal to the energy value of the point. However, when a configuration has not yet reached a stable state, the energies in the local moonscape will differ. The point will tend to move to those areas on the moonscape that have lower energies.

The moonscape is drawn as a shaded mesh. Height and color are used to indicate when the energy is higher (green) or lower (red) than the energy in the point.

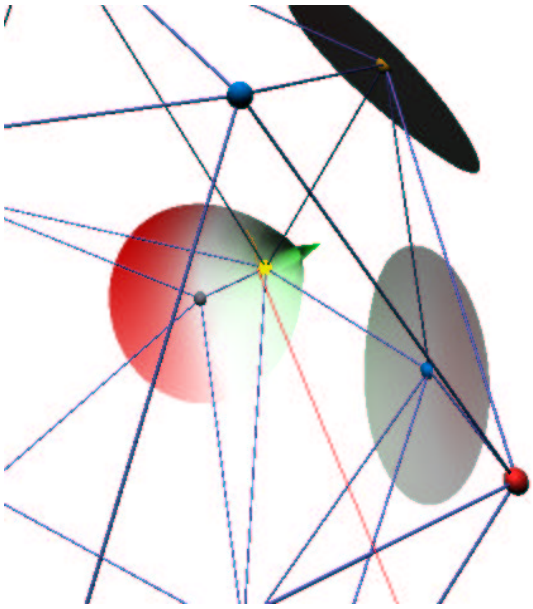

Figure 2. Local moonscape around a point. The moonscape visualizes the stability around the point.

4.2. Interaction

Users may perturb a configuration by dragging a selected point to a new position. During this interaction the configuration will not be updated. Only when the interaction is finished (defined by releasing the selected point), will the computation will immediately continue with the perturbed configuration. In this way the user can gain insight into the stability of a configuration; i.e. the perturbed configuration may converge to the previous configuration or it may converge to a different configuration.

In addition to the simple method of perturbing a configuration, we have added a technique that we call *snap convergence*. This technique computes and displays the converged configuration *while* the user is perturbing a configuration. Figure 3 illustrates snap convergence. A selected point (indicated by the yellow sphere and a red line as a pointer from the wand) in the current configuration is being dragged to a new location. The upper left hand corner of the image shows a list of stable configurations that have been encountered. One configuration in the list is highlighted (in yellow) to indicate the configuration which the current configuration will converge to if the selected point is to be released. When the user drags the point to a new location a different configuration may be highlighted.

Also, a history of red or yellow crosses are drawn on the path taken by the dragged point. The color of the cross indicates the configuration which will be converged too at that location. In this way, a user can interactively construct colored regions which denote the energy surface.

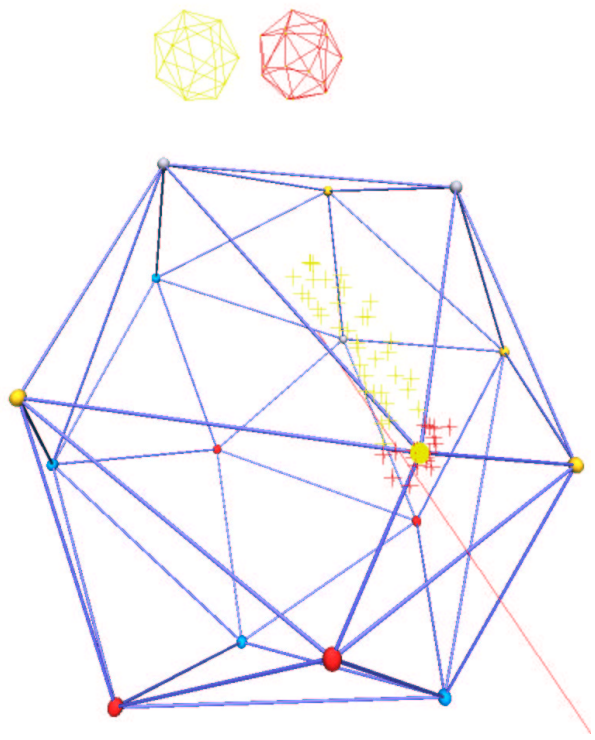

Figure 3. Snap convergence during perturbation of a configuration. The small red configurations on the upper left are a list of converged configurations. The yellow configuration is the configuration to which the current configuration will converge. Small yellow and red crosses are drawn on the path taken by the dragged point to indicate the stable configuration at the location of the cross.

Snap convergence can be used to gain insight into how much a configuration must be perturbed before it will converge to a different configuration. Snap convergence annotated with colored crosses can be used to provide an indication of the global energy landscape.

4.3. Performance Issues

The Fekete environment is implemented on top of PVR, a modular library for developing portable VR applications, [14]. PVR provides support for multiple processing, allowing the rendering module(s) to be decoupled from the computational and device management modules.

The goal was to implement the Fekete virtual environment which would achieve an update rate of minimally 10 frames/sec. Time critical computational algorithms have been essential for obtaining this responsiveness. We describe some computational issues that are performance critical, and discuss tradeoffs that have been made to realize the performance goals:

- computation

  Solving the system (3) by standard differential-algebraic equations software works well in that these packages find larger values for $f^*(n)$ in less computing time than off-the-shelf optimization packages. The algorithm given in section 3.4, however, performs better than standard differential-algebraic equations solvers by orders of magnitude. For example, for a random initial configuration and a relatively small number of points, the algorithm can find a stable configuration in a few seconds.

  In addition, the algorithm is, as far as we know, the most robust way to solve the Fekete problem. We rely on its robustness property when perturbing a configuration.

- local moonscapes

  Local moonscapes define the energy values in a neighborhood of a point. A moonscape is defined as a mesh, parameterized with its radius, and the number of rings R it contains. Each ring r adds 2^{r+1} points to the mesh. Each subsequent ring r is connected to ring $r-1$ by a triangle strip of $2^r \times 3$ triangles. The first ring defines a triangle fan of four triangles with the particle as the center point. Hence, $2^{R+2} - 4$ additional energy computations are required to compute the moonscape.

  Increasing the number of rings will result in a more accurate representation of the energy in the neighborhood of a point. The trade-off is performance versus accuracy. We have chosen to adaptively decrease the accuracy when a configuration is not yet in a stable state; i.e., when the animation is (quickly) changing. When the configuration converges, the accuracy of the moonscape will increase.

- snap convergence

  A configuration is perturbed by dragging a point to a new location. Snap convergence does an on-the-fly convergence computation of a configuration while it is being perturbed. For each new location a new converged configuration will need to be computed. Snap convergence is realized by running the computation in

time-critical mode. It is important that solver can compute a converged configuration a few times per second.

In order to achieve this, the solver has been adapted in three ways. First, the time step parameter has been relaxed. This allows larger time steps to be taken, resulting in a faster computations. However, in some cases, larger time steps may compromise the accuracy and robustness of the computation. Second, the tolerance used in comparing rotation free canonical forms is relaxed, so that similar configurations can be found faster. Third, the number of iterations needed to converge to a solution is bounded to an upper limit. This allows the computation to always return in a maximal number of iterations.

The techniques used to speed up snap convergence compromise the computation, in that the correct converged configuration is found. This is a trade-off between computational accuracy and interactive performance. It is also possible that a converged configuration may not be found within the time constraints posed by snap convergence. In this case, no configuration will be highlighted.

5. Discussion

The virtual environment has been used to study the dynamics of the alternative formulation of the Fekete problem. In Table 1 we list some important notions that provide insight into the mathematics of the model (left) and the corresponding interactive visualization techniques used to display these notions. For example, the convex hull is a very useful technique to gain insight into regularities and symmetries of a configuration. Also, combining a moonscape with dragging and animating the solution of a configuration gives insight into the stability and sensitivity to perturbation of a configuration.

5.1. Why virtual environments?

The remaining question is: what is the added value of a virtual environment for studying the Fekete problem?

First, we list some concrete benefits we found:

- Motion parallax (obtained via head coupling with the display) has proved to be an essential visual depth cue to interpreting the convex hull. Users can search for regularities in configurations by simply moving the head to a different location. It is doubtful that this effect can be achieved via non-head coupled displays by using 3D graphics rotation techniques, such as "rocking" or variations of "grand tour".

insight to problem	**visualization techniques**
stability	moonscape
regularities/symmetry	convex hull
velocity of points	trajectories, animation
interactions between points	hull, moonscape, animation
surface count	hull
convergence paths	trajectory
sensitivity to perturbation	dragging + animation
global energy landscape	snap convergence

Table 1. Notions that provide insight to the Fekete problem (left) and their corresponding interactive visualization techniques (right). Motion parallax, spatial input and interactive performance are essential ingredients to provide a natural interface for the visualization techniques.

- Spatial input is used when perturbing a configuration. Dragging a point with spatial input devices has proven to be more intuitive, efficient, and accurate than with two dimensional input devices.

- Other advantages of using for a virtual environments are the additional depth cues when interpreting trajectories and small local disturbances on a moonscape.

Also, a number of more general features can be identified:

- Due to the more "natural" environment, the user can perceive and interpret visualization cues more rapidly. For example, by taking advantage of motion parallax it was possible to observe the slow convergence of a number of points to their equilibrium locations in a plane, by moving the observation point to a location on the plane. This was done quite subconsciously by the observer, and is something which would probably never have been noticed using more conventional visualization techniques.

- Due to the intuitive representation of the dynamics of configurations, more information can be represented in the interface. For example, the history of a configuration (i.e. trajectories), the current configuration (i.e. hull) and the neighborhood of a point (i.e. moonscapes) can be shown in one image. It is difficult to provide his amount of information on a 2D display due to, among others, hidden lines and surfaces.

- The fast and intuitive interaction invites the user to experiment and explore the dynamics of a configuration.

5.2. Evaluation

The implemented virtual environment is being used extensively. Users find that the environment provides a more natural interface, allowing them to perceive, interpret and interact with point configurations more rapidly. The explorative nature environment of the environment allows users to see relationships and test hypotheses of configurations in detail. In addition, various pathological configurations that converge very slowly could be analyzed.

The environment is used in two ways. First, as discussed above, it allowed users to gain insight into the complex dynamics of the Fekete problem. Second, the environment was used for the development and validation of the model and the solver themselves. Developers were able to test and debug many solving strategies, in particular, how to deal with variable stepsizes during integration.

Nevertheless, there are also a number of drawbacks of this environment. First, the current interface contains only minimal quantitative information. Quantitative values such as point positions and velocities, plots of energy history, etc are absent. Providing this information in a fixed area in the workspace would not be an adequate solution, since the user would need to switch viewing directions to examine this information. Second, the development of the interface was not done by the same people that developed the numerical codes. Programming VR interfaces is still very cumbersome for experienced programmers that do not have the required skills.

6. Conclusion

In this paper we discussed a virtual environment that allows the user to analyze the dynamics of Fekete point configurations. We found that a highly interactive environment, combined with a time-critical computation, provides valuable insight into the symmetry, stability and sensitivity of the point configurations. We believe that the virtual environment provides a more natural interface to complex problems, allowing users to perceive, interpret and interact with the problem more rapidly.

References

[1] M. Fekete. Über die Verteilung der Wurzeln bei gewisser algebraischen Gleichungen mit ganzzahligen Koeffizienten. *Mathematische Zeitschrift*, 17, 1923.

[2] P.M. Pardalos. An open global optimization problem on the unit sphere. *Journal of Global Optimization*, 6:213, 1995.

[3] T. Erber and G.M. Hockney. Complex systems: Equilibrium configurations of n equal charges on a sphere. *Advances in Chemical Physics*, XCVII:495–594, 1997. ISBN 0-471-16285-X.

[4] G. Francis, J.M Sullivan, R.B. Kusner, K.A. Brakke, C. Hartman, and G. Chappell. The minimax sphere eversion. In H.C. Hege and K. Polthier, editors, *Visualization and Mathematics*, pages 3–20. Springer, 1997.

[5] D. Roseman. What should a surface in 4-space look like? In H.C. Hege and K. Polthier, editors, *Visualization and Mathematics*, pages 67–82. Springer, 1997.

[6] S. Bryson. Real-time exploratory scientific visualization and virtual reality. In L.J. Rosenblum et al, editor, *Scientific Visualization: Advances and Challenges*, pages 65–86. Academic press, 1994.

[7] S.T. Bryson and S. Johan. Time management, simultaneity and time-critical computation in interactive unsteady visualization environments. In R. Yagel and G.M. Nielson, editors, *Proceedings IEEE Visualization '96*, pages 255–262. IEEE Computer Society Press, 1996.

[8] S. Bryson. Time-critical computational algorithms for particle advection in flow visualization. In *Visualization '90 (Proceedings of the IEEE 1990 Visualization Conference, Late Breaking Hot Topics)*, 1999.

[9] W.J.H. Stortelder, J.J.B. de Swart, and J.D. Pintér. Finding elliptic Fekete point sets: Two numerical solution approaches. *CWI Quarterly*, 12(1):63–76, 1999.

[10] M. Shub and S. Smale. Complexity of Bezout's theorem III. Condition number and packing. *Journal of Complexity*, 9:4–14, 1993.

[11] J. Frank and B. Leimkuhler. Dissipated symplectic methods for constrained minimization, 1999. In preparation.

[12] J.M. Sanz-Serna and M.P. Calvo. *Numerical Hamiltonian Problems*. Chapman & Hall, 1994.

[13] W. Huang and B. Leimkuhler. The adaptive Verlet method. *SIAM Journal on Scientific Computing*, 18:239–256, 1997.

[14] R. van Liere and J.D. Mulder. PVR - an architecture for portable VR applications. In M. Gervautz, A. Hildebrand, and D. Schmalstieg, editors, *Virtual Environments '99, Proceedings of the Virtual Environments Conference & 5th Eurographics Workshop*, pages 125–135. Springer Verlag, 1999.