Coarse to Fine and Back Again

Irad Yavneh

Department of Computer Science

Technion – Israel Institute of Technology

irad@cs.technion.ac.il

Some Relevant Books:

1. W.L. Briggs, V.E. Henson, and S.F. McCormick: "A Multigrid Tutorial", 2nd ed., SIAM, 2000.
2. U. Trottenberg, C.W. Oosterlee, and A. Schueller: "Multigrid", Academic Press, 2001.
3. Brandt, A., "1984 Guide with Applications to Fluid Dynamics", GMD-Studie Nr. 85, 1984.
4. Hackbusch, W., "Multigrid Methods and Applications", Springer, Berlin, 1985.
5. W. Hackbusch and U Trottenberg eds.: "Multigrid Methods", Springer-Verlag, Berlin, 1982.
6. Wienands, R., and Joppich, W., "Practical Fourier Analysis for Multigrid Methods", Chapman & Hall/CRC, 2004.

What's it about?

A framework of efficient iterative methods for solving problems with many variables and many scales.

- Framework: common concept, different methods.
- Efficient: usually $O(N)$ or $O(N \log N)$ operations
  The importance of efficient methods becomes greater as computers grow stronger!
- Iterative: most nontrivial problems in our field cannot be solved directly efficiently.
- Solving: approximately, subject to appropriate convergence criteria, constraints, etc.
- Many variables: the larger the number of variables, the greater the gain of efficient methods.
- Many scales: typical spatial and/or temporal sizes.

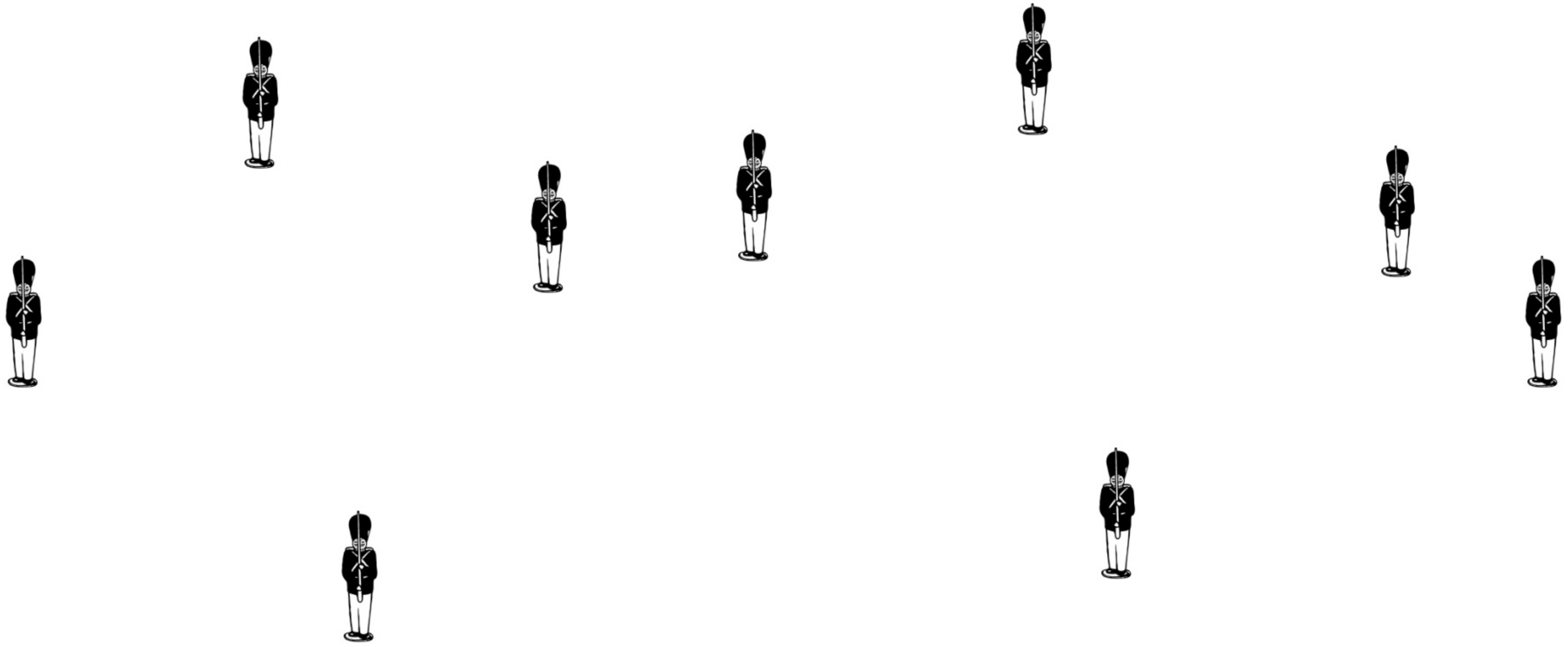
A framework of efficient iterative methods for solving problems with many variables and many scales.

Basic Concepts: Local vs. Global processing.

Imagine a large number of soldiers who need to be arranged in a straight line and at equal distances from each other.

The two soldiers at the ends of the line are fixed.

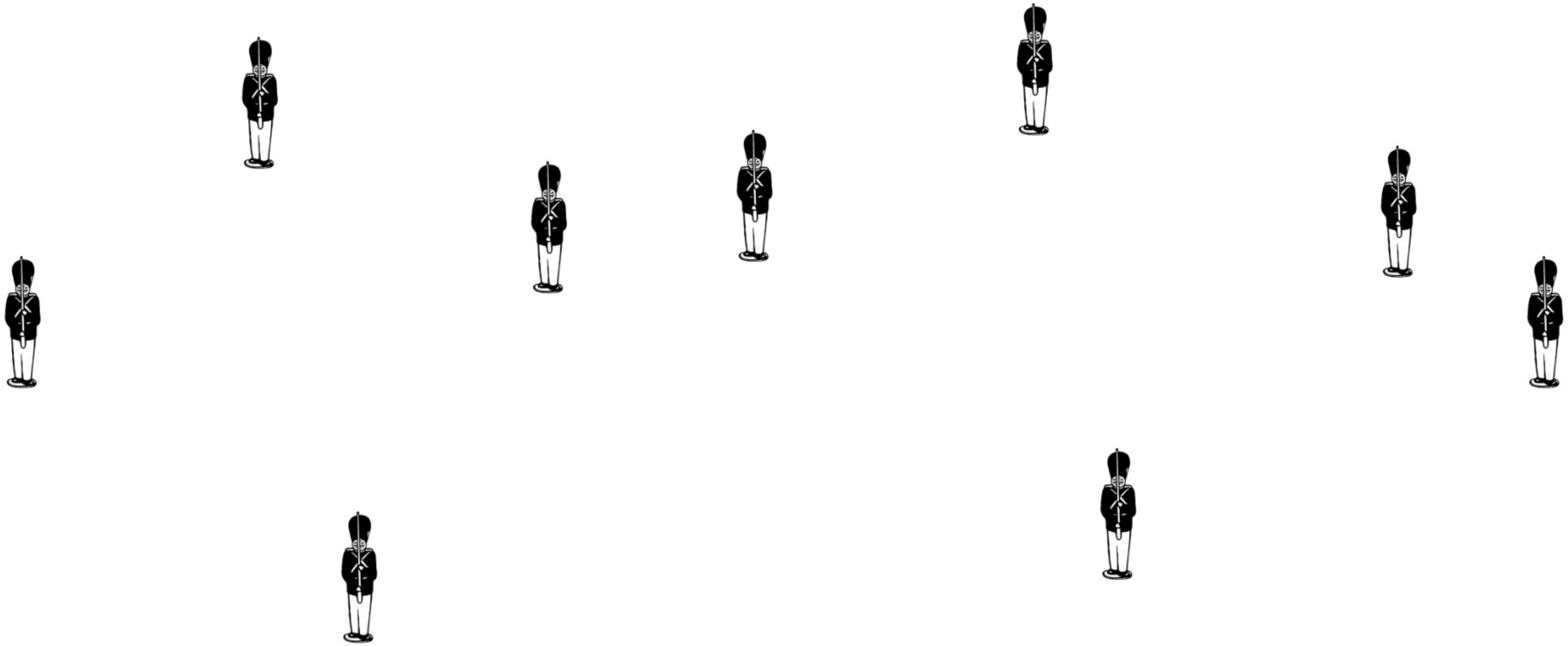
We number the soldiers 0 to N , and the length of the entire line is L.



Initial Position

Final Position

Global processing. Let soldier number j stand on the line connecting soldier 0 to soldier N at a distance jL/N from soldier number 0.

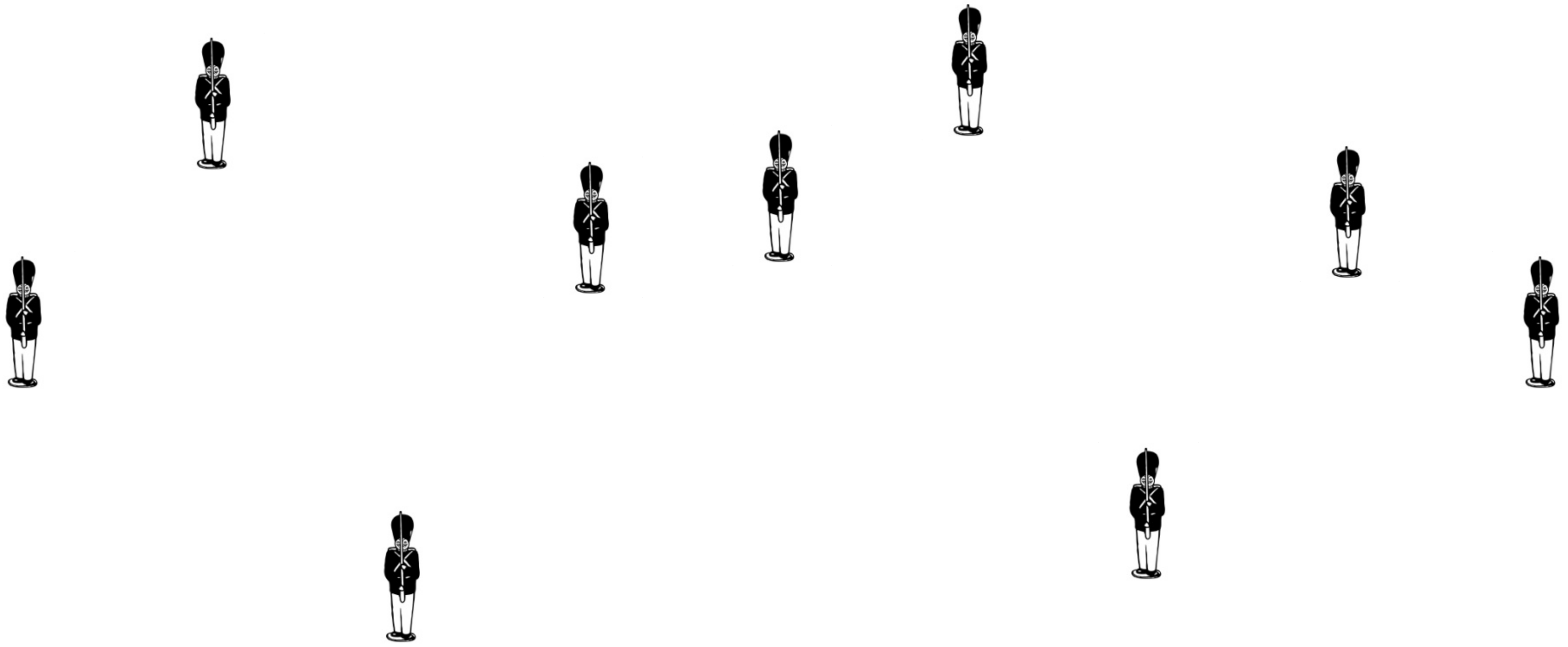


This method solves the problem directly, but it requires a high degree of sophistication: recognition of the extreme soldiers and some pretty fancy arithmetic.

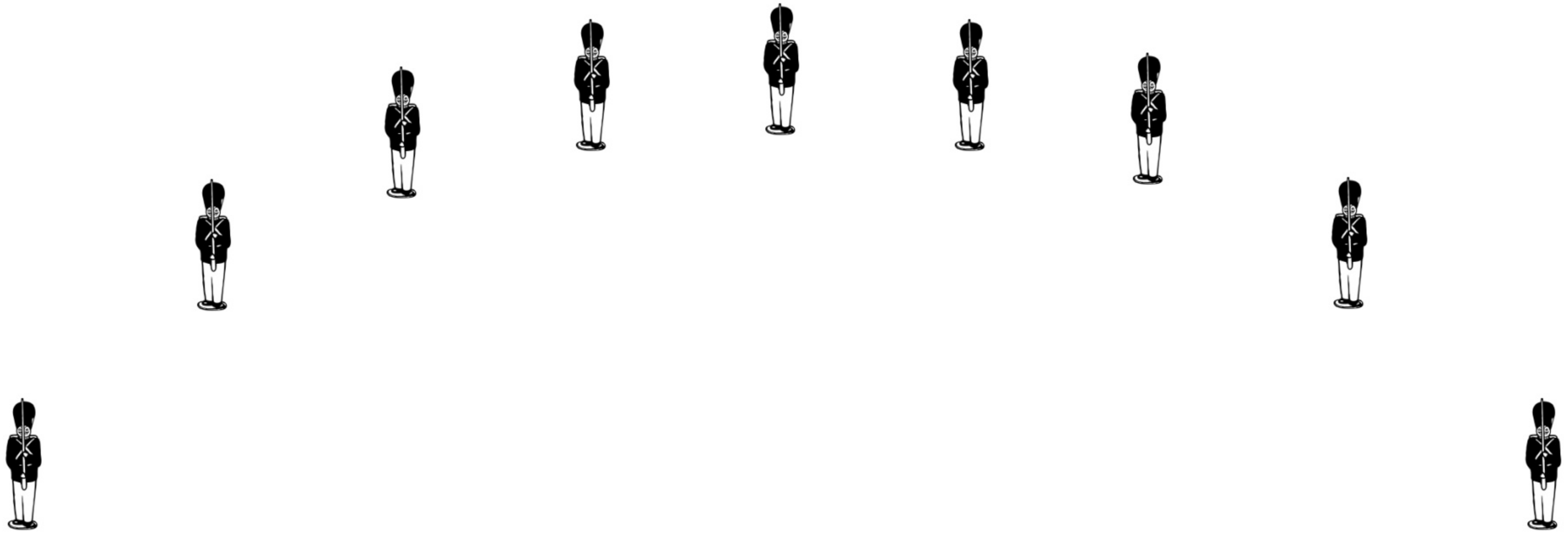
Local processing (iterative method). Suppose that the inner soldiers' initial position is $\mathbf{x}^{(0)} = (x_1, x_2, \ldots, x_{N-1})^{(0)}$. Then repeat for i=1,2,…: Let each soldier j, j=1,…N-1 at iteration i move to the point midway between the locations of soldier j-1 and soldier j+1 at iteration i-1:

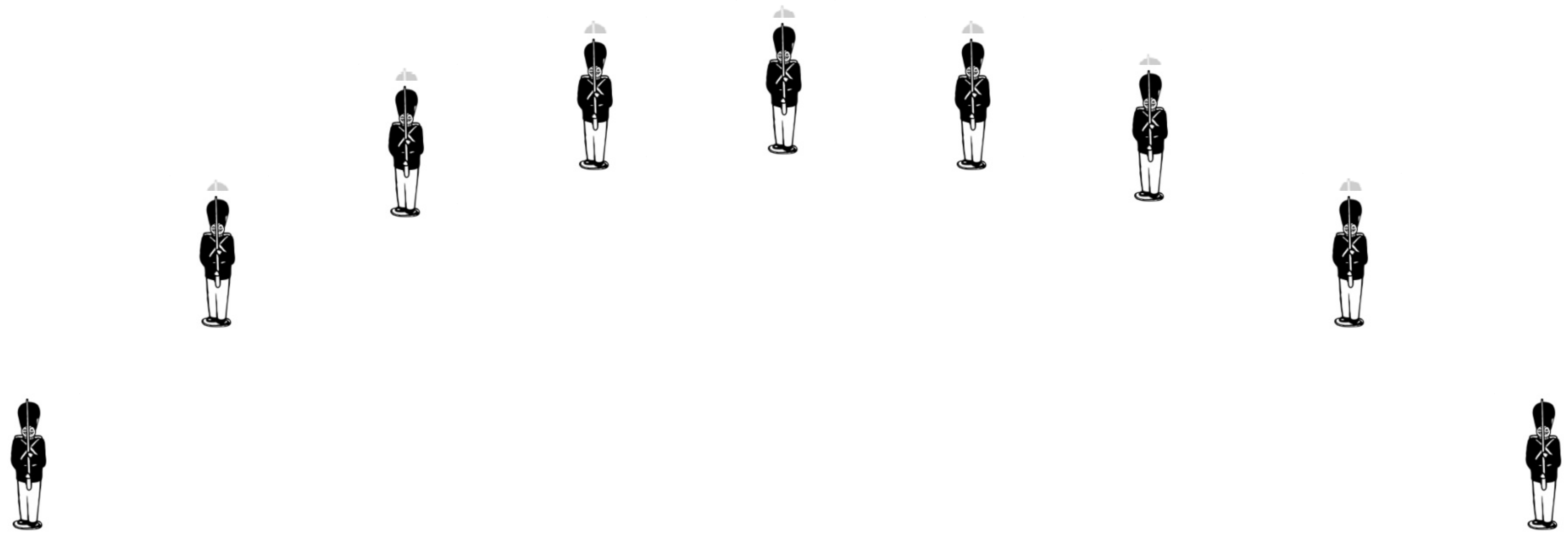
$$x_j^{(i)} = \frac{1}{2}\left(x_{j-1}^{(i-1)} + x_{j+1}^{(i-1)}\right)$$

This is an iterative process. Each iteration brings us closer to the solution(?). The arithmetic is trivial.

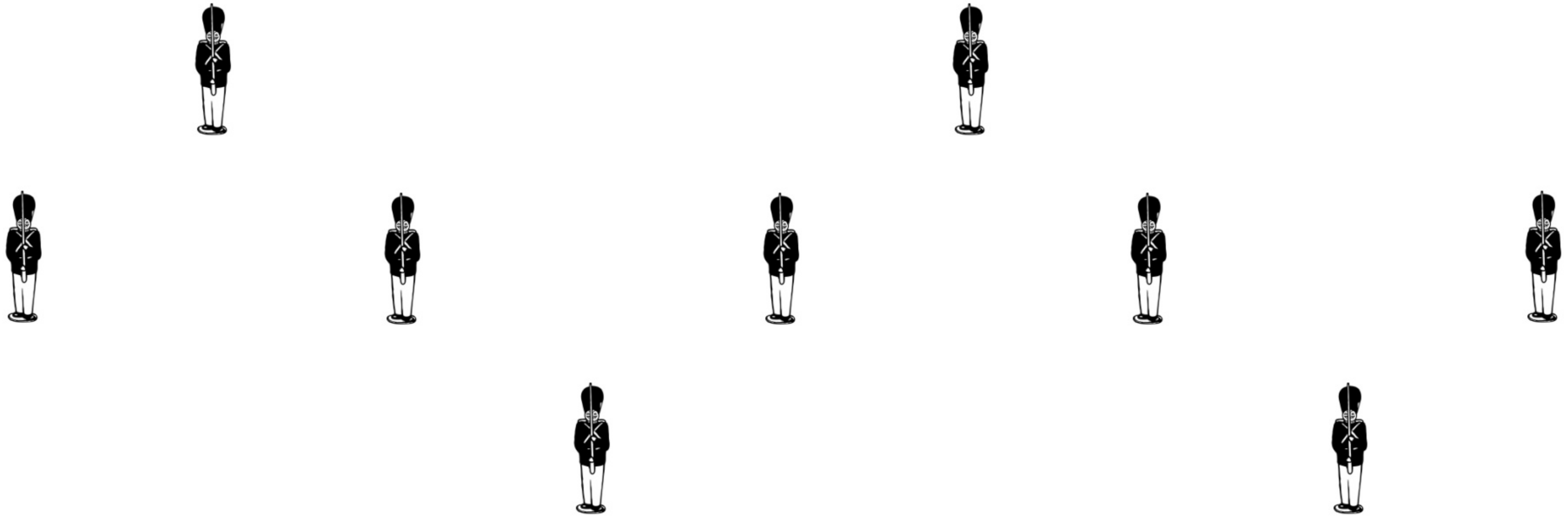


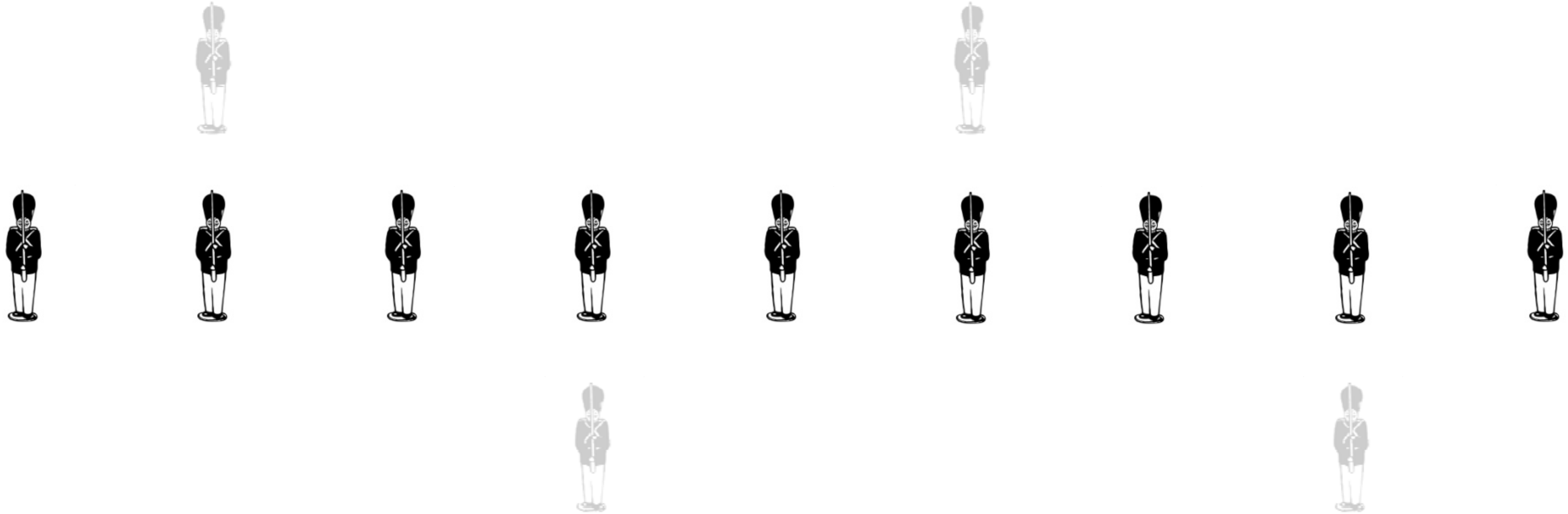
A step in the right direction



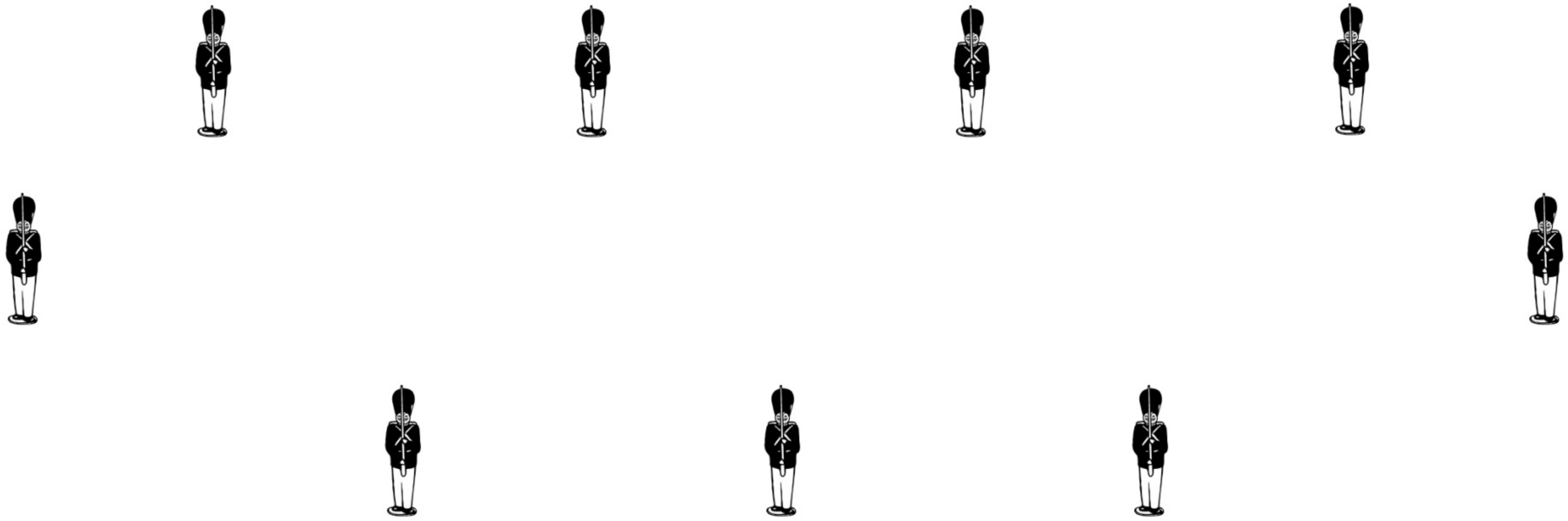


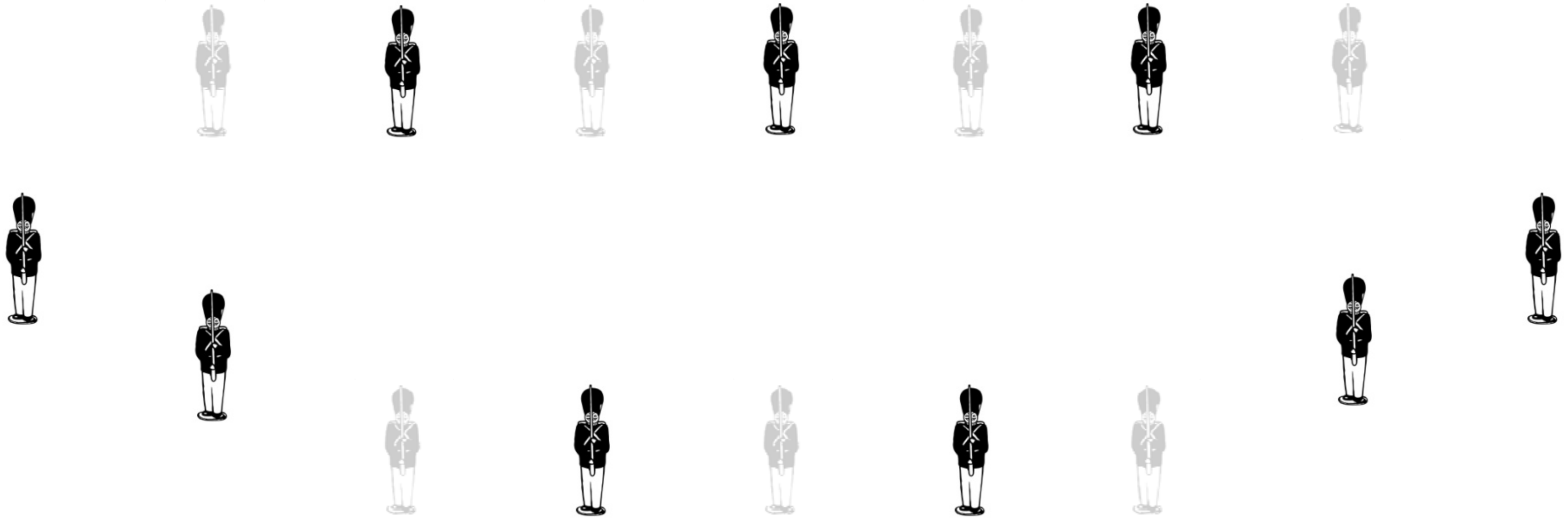
Slow convergence



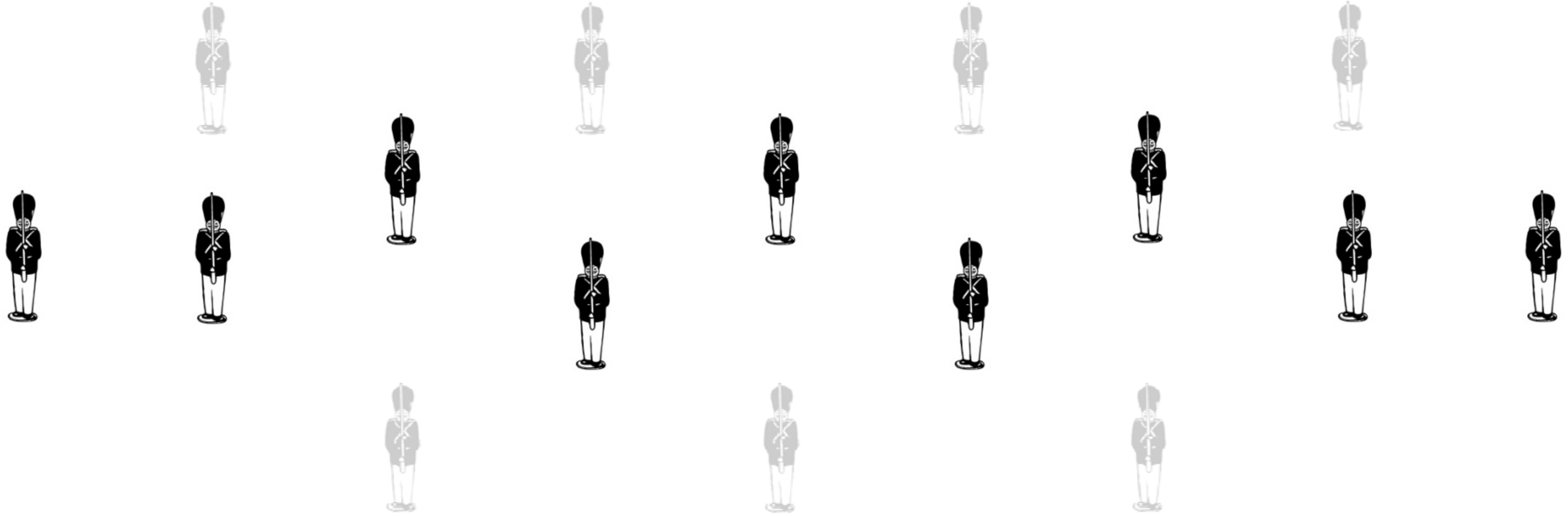


Fast convergence

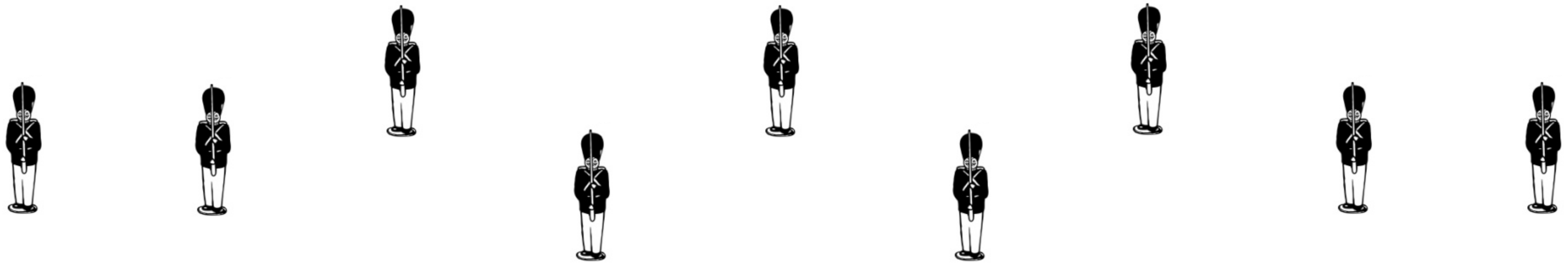




Slow convergence



Local solution: damping

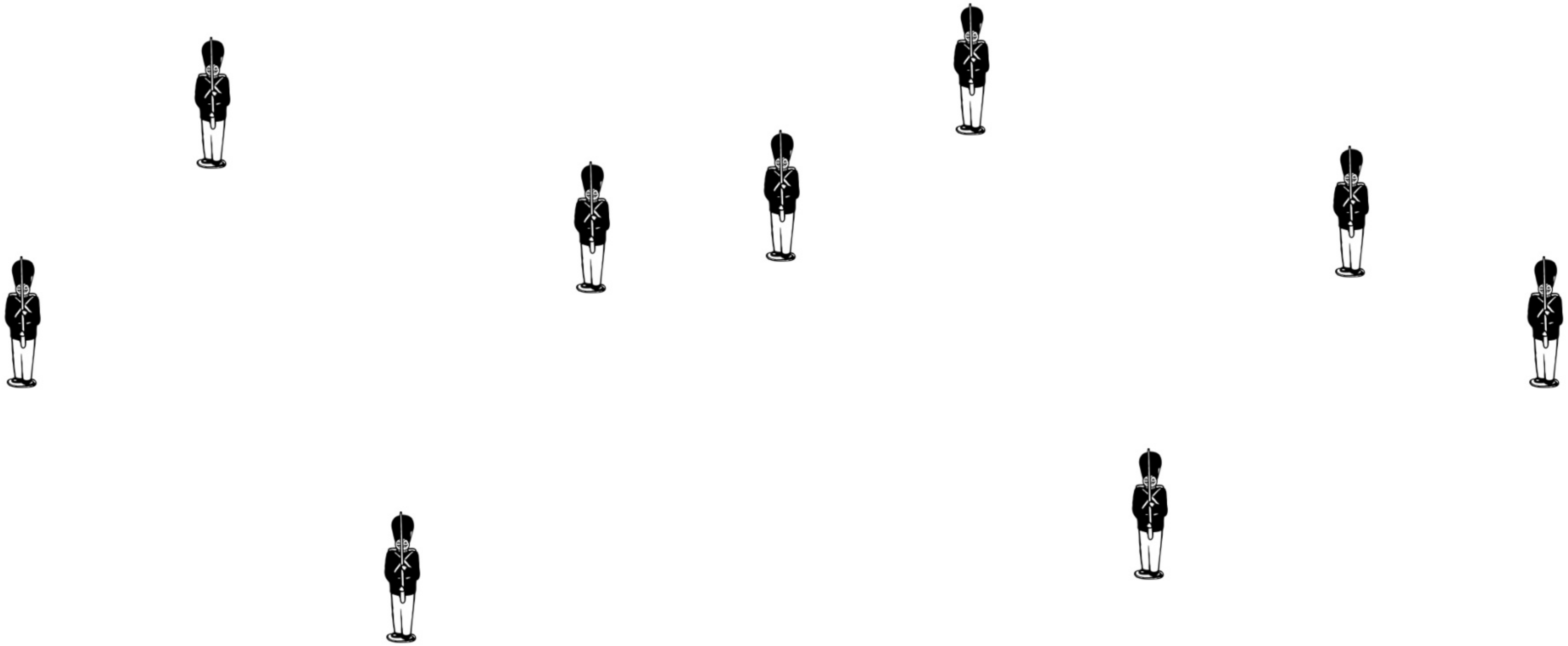


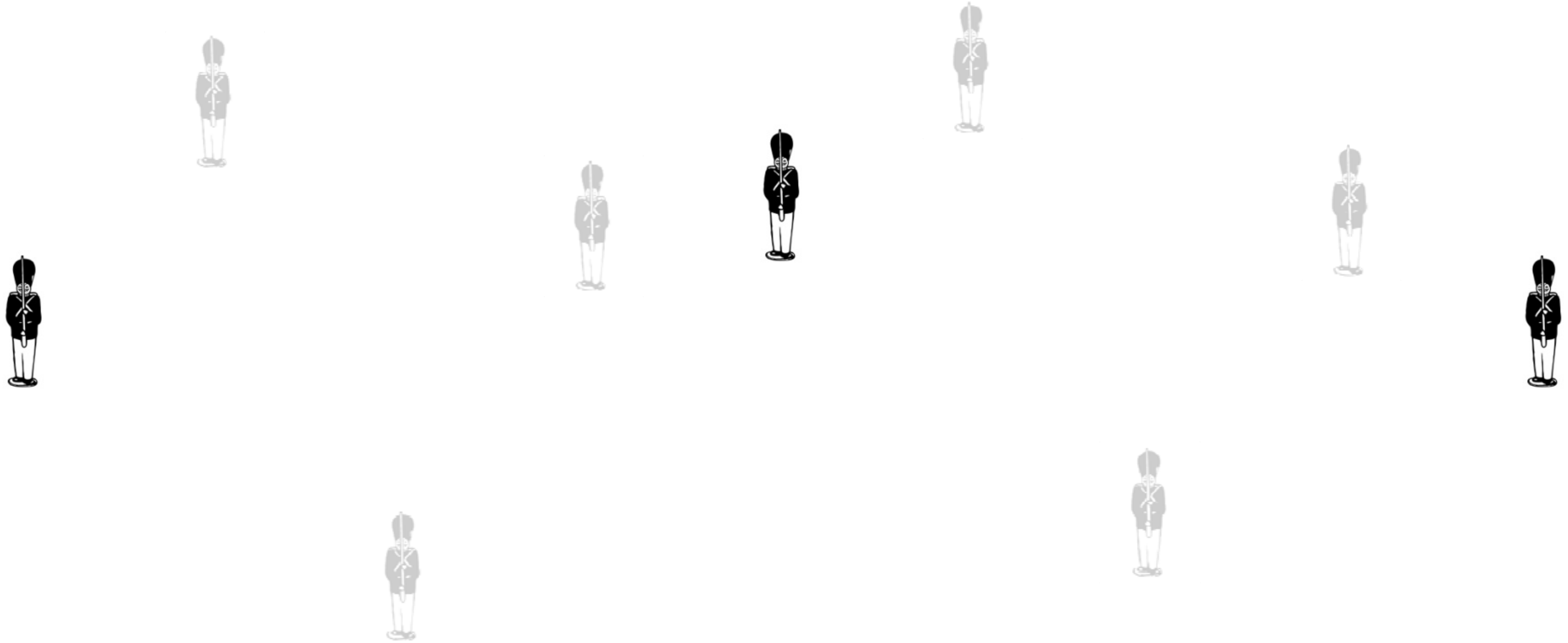
Local solution: damping

Local solution: damping

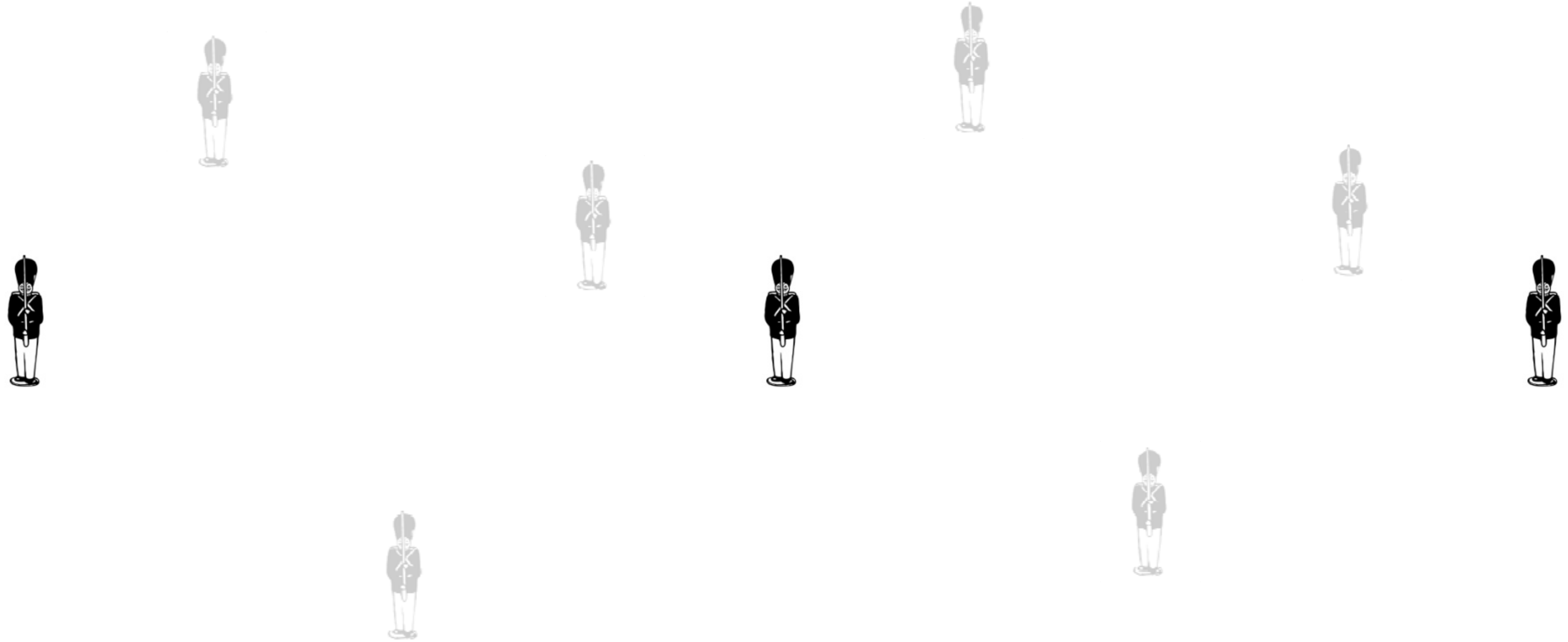
Local solution: damping

The multiscale idea: Employ the local processing with simple arithmetic. But do this on all the different scales.

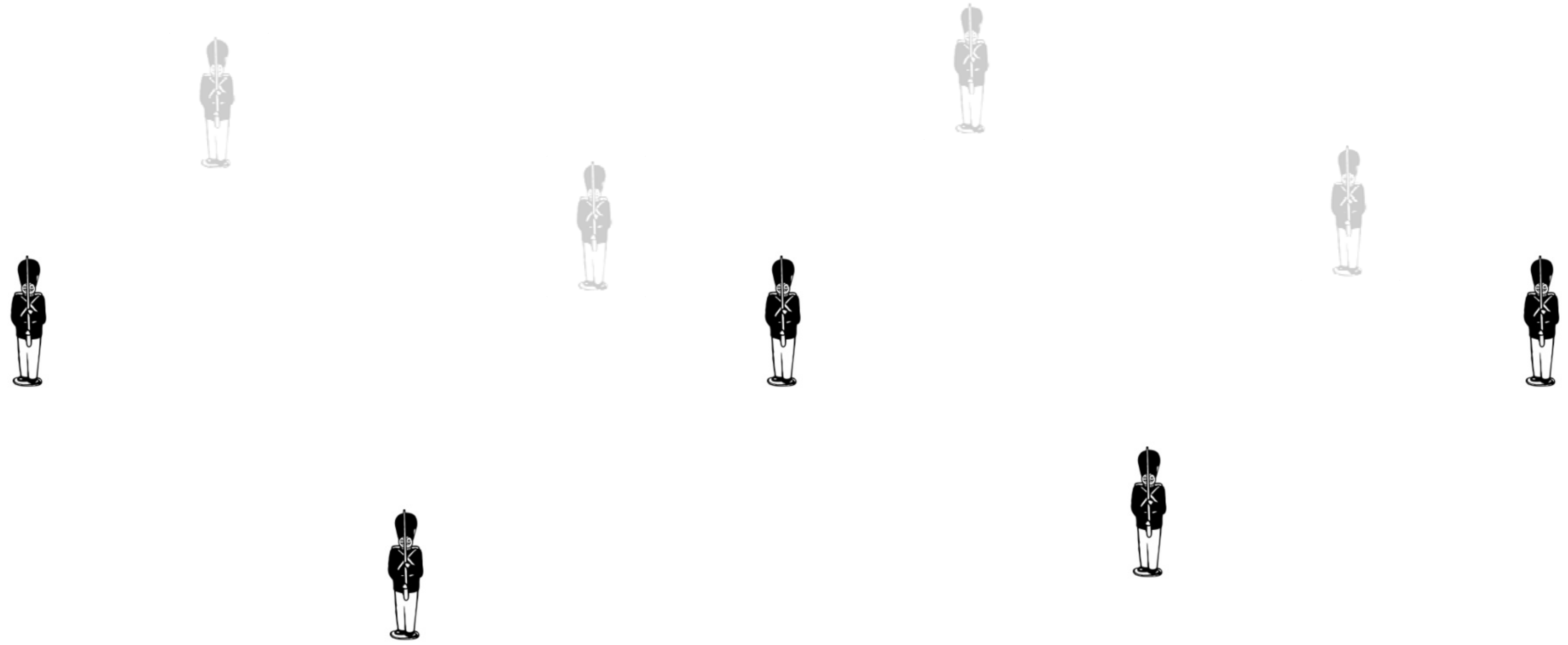




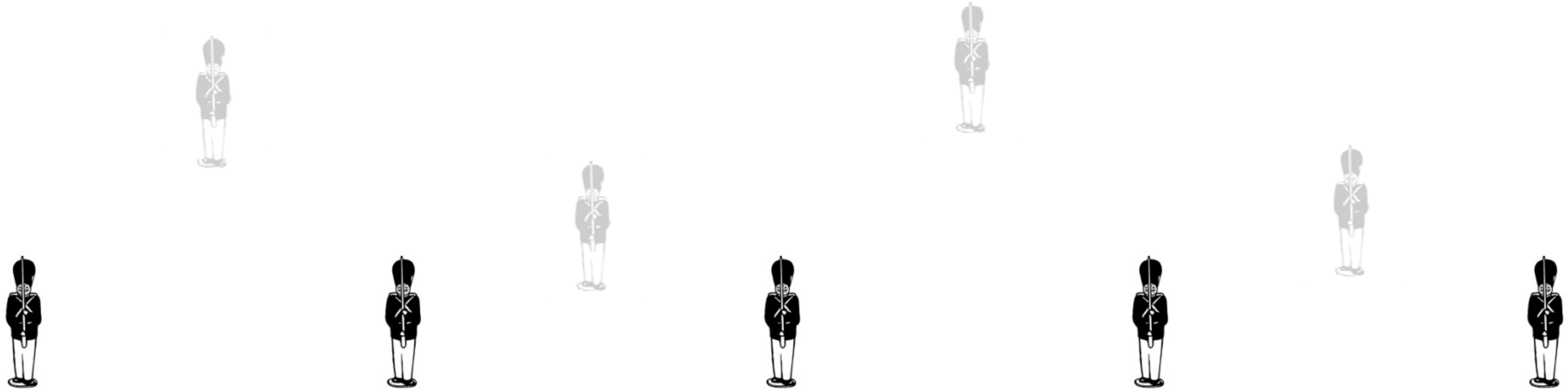
Large scale



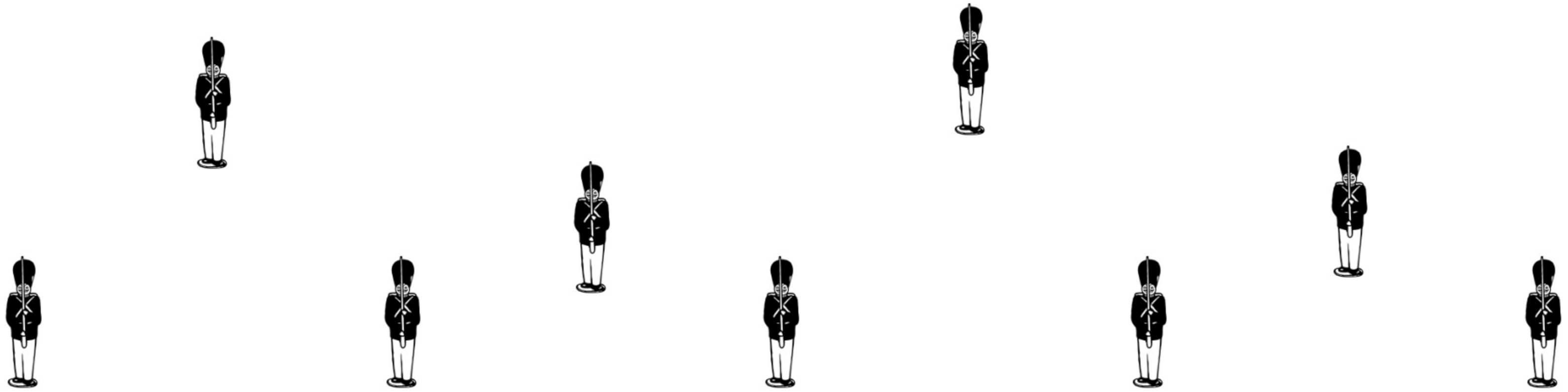
Large scale



Intermediate scale



Intermediate scale



Small scale

HOW MUCH DO WE SAVE?

Analysis of the Jacobi iterative process

Matrix representation:

$$\mathbf{x}^{(i)} = \mathbf{S}\mathbf{x}^{(i-1)}$$

where

$$\mathbf{S} = \frac{1}{2}\begin{bmatrix} 0 & 1 & & & & & \\ 1 & 0 & 1 & & & & \\ & 1 & 0 & 1 & & & \\ & & \cdots & \cdots & \cdots & & \\ & & & 1 & 0 & 1 & \\ & & & & 1 & 0 & 1 \\ & & & & & 1 & 0 \end{bmatrix}$$

This matrix **S** has N - 1 linearly independent eigenvectors, $\mathbf{v}^k$, and corresponding real eigenvalues, λ_k

$$\mathbf{S}\,\mathbf{v}^k = \lambda_k \mathbf{v}^k .$$

Since $\mathbf{v}^k$ span the space $\Re^{N-1}$, any initial configuration of the soldiers can be written as a linear combination:

$$\mathbf{x}^{(0)} = \sum_{k=1}^{N-1} c_k \mathbf{v}^k$$

with some coefficients, c_k.

Hence, we obtain after m iterations:

$$\mathbf{x}^{(m)} = \mathbf{S}\mathbf{x}^{(m-1)} = \mathbf{S}^2\mathbf{x}^{(m-2)} =$$

$$\ldots = \mathbf{S}^m\mathbf{x}^{(0)} = \mathbf{S}^m \sum_k c_k \mathbf{v}^k = \sum_k c_k \lambda_k^m \mathbf{v}^k$$

Conclusion:

$$\lim_{m\to\infty} \mathbf{x}^{(m)} \to 0 \quad \textit{if} \quad |\lambda_k| < 1, \quad k = 1, \ldots, N-1$$

The iteration converges if the spectral radius, ρ, of the iteration matrix, $\mathbf{S}$, is smaller than 1.

Observation: the eigenvectors and eigenvalues of the matrix $\mathbf{S}$ are given by

$$\mathbf{v}^k = \left\{ \mathbf{v}_j^k \right\} = \sin\left(\frac{jk\pi}{N} \right), \quad j = 1, \ldots, N-1,$$

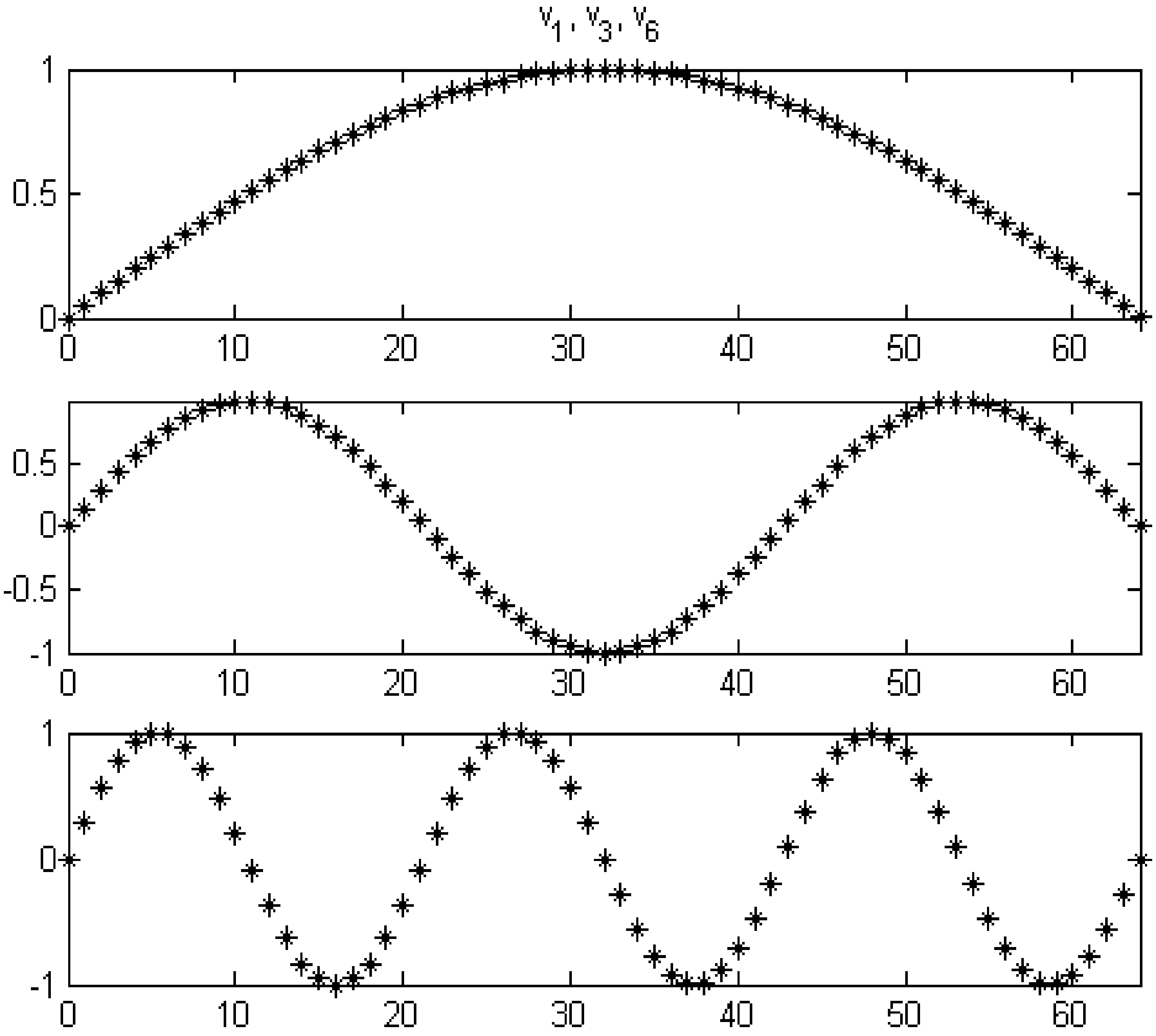
$$\lambda_k = \cos\left(\frac{k\pi}{N} \right),$$

with $k = 1, \ldots, N-1.$

Proof: Using the trigonometric identity,

$$\frac{1}{2}\left[\sin\frac{(j-1)\,k\pi}{N} + \sin\frac{(j+1)\,k\pi}{N} \right] = \cos\frac{k\pi}{N} \sin\frac{jk\pi}{N},$$

and the fact that $\sin 0 = \sin \pi = 0$, we obtain by substitution, $\mathbf{S}\,\mathbf{v}^k = \lambda_k \mathbf{v}^k$.


v_1 , v_3 , v_6
1
0.5
0
0
10
20
30
40
50
60
0.5
0
-0.5
-1
0
10
20
30
40
50
60
1
0
-1
0
10
20
30
40
50
60

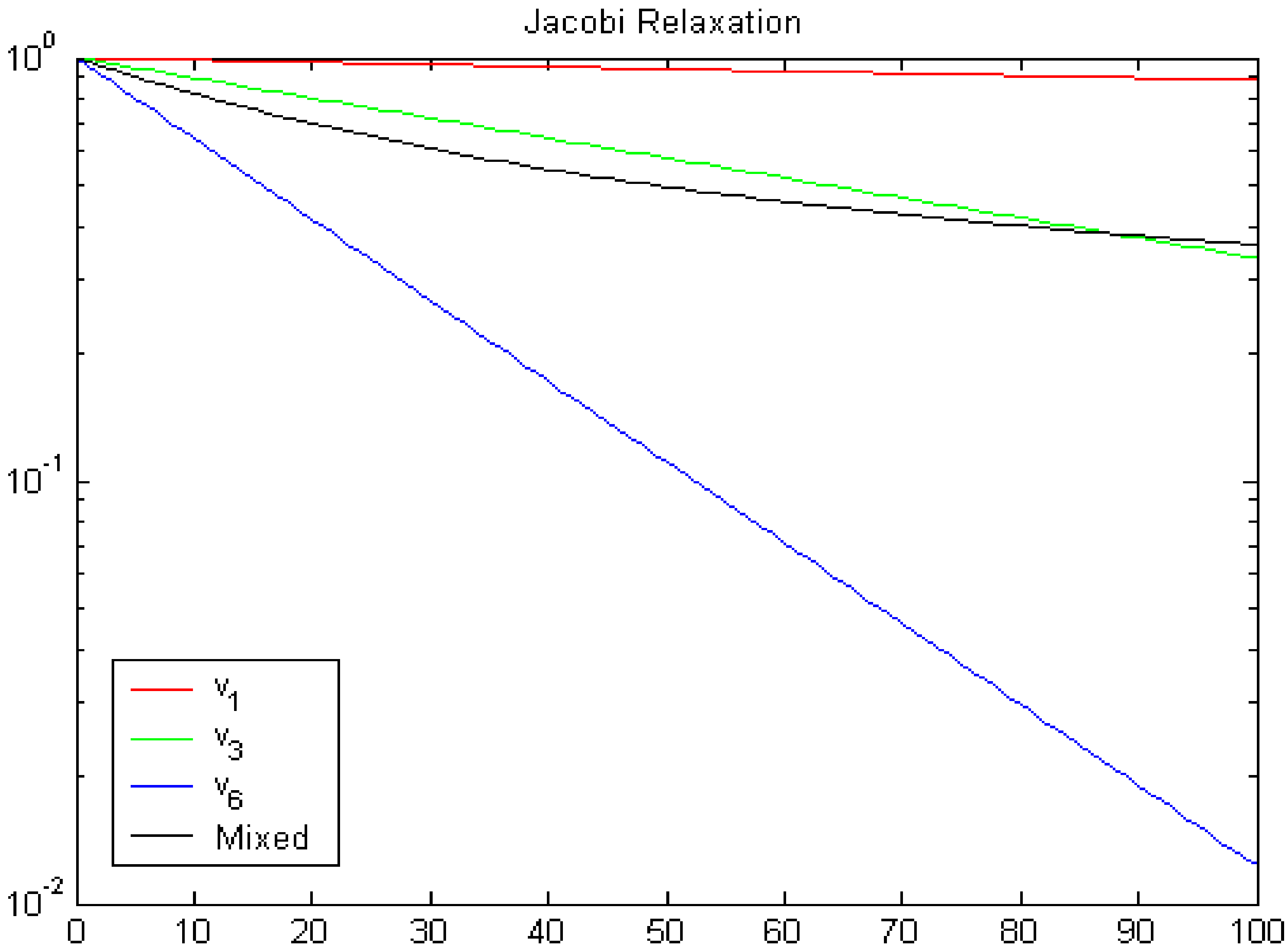
Note: since $|\lambda_k| < 1$, the method converges. But, for some eigenvectors, $|\lambda_k|$ is close to 1, so convergence is slow. In particular, for $k\pi/N \ll 1$, we have,

$$\lambda_k = \cos\left(\frac{k\pi}{N}\right) \approx 1 - \frac{1}{2}\left(\frac{k\pi}{N}\right)^2 .$$

For $k = 1$ we obtain

$$\lambda_1^m \approx \left[1 - \frac{1}{2}\left(\frac{\pi}{N}\right)^2\right]^m \approx e^{-\frac{1}{2}m\left(\frac{\pi}{N}\right)^2} .$$

Conclusion: $O(N^2)$ iterations are required to reduce such an error by an order of magnitude.


Jacobi Relaxation
10^0
10^{-1}
10^{-2}
0
10
20
30
40
50
60
70
80
90
100
v_1
v_3
v_6
Mixed

How much work do we save?

Jacobi's method requires about N^2 iterations and $N^2 * N = N^3$ operations to improve the accuracy by an order of magnitude.

The multiscale approach solves the problem in about $\mathbf{Log_2(N)}$ iterations (whistle blows) and only about N operations.

Example: for $N = 1000$ we require about:

10 iterations and 1000 operations

instead of about

1,000,000 iterations and 1,000,000,000 operations

How important is computational efficiency?

Suppose that we have three different algorithms for a given problem, with different computational complexities for input size N :

Algorithm 1: $10^6 N$ operations

Algorithm 2: $10^3 N^2$ operations

Algorithm 3: N^3 operations

Suppose that the problem size, N, is such that Algorithm 1 requires one second.

How long do the others require?

Computer Speed (ops/sec)	*N*	Algorithm 1 $O(N)$	Algorithm 2 $O(N^2)$	Algorithm 3 $O(N^3)$
1M ($\sim 10^6$) (1980's)	1	1 sec	0.001 sec	0.000001 sec
1G ($\sim 10^9$) (1990's)	1K	1 sec	1 sec	1 sec
1T ($\sim 10^{12}$) (2000's)	1M	1 sec	17 min	12 days
1P ($\sim 10^{15}$) (2010's)	1G	1 sec	12 days	31,710 years

Stronger Computers $\Rightarrow$

Greater Advantage of Efficient Algorithms!

The catch: in less trivial problems, we cannot construct appropriate equations on the large scales without first propagating information from the small scales.

Skill in developing efficient multilevel algorithms is required for:

1. Choosing a good local iteration.

2. Choosing appropriate coarse-scale variables.

3. Choosing inter-scale transfer operators.

4. Constructing coarse-scale approximations to the fine-scale problem.

Damping

Recall: the eigenvectors and eigenvalues of the iteration matrix $\mathbf{S}$ are given by

$$\mathbf{v}^k = \{v_j^k\} = \sin\left(\frac{jk\pi}{N}\right), \quad j = 1, \ldots, N-1,$$

$$\lambda_k = \cos\left(\frac{k\pi}{N}\right),$$

with $k = 1, \ldots, N-1$.

Note that convergence is also slow for $\boldsymbol{k / N \approx 1}$.

This slow convergence can be overcome by damping:

$$x_j^{(i)} = (1-\omega)x_j^{(i-1)} + \omega\frac{1}{2}\left(x_{j-1}^{(i-1)} + x_{j+1}^{(i-1)}\right),$$

where ω is a parameter.

Then, $\mathbf{x}^{(i)} = \mathbf{S}_\omega \mathbf{x}^{(i-1)}$, where

$$\mathbf{S}_\omega = (1-\omega)\mathbf{I} + \omega\mathbf{S}.$$

Note: $\mathbf{v}^k$ are eigenvectors of $\mathbf{S}_\omega$. The corresponding eigenvalues are now $\lambda_k^{(\omega)} = 1-\omega+\omega\lambda_k = 1-\omega(1-\lambda_k)$.

For $0 < \omega \leq 1$, we have convergence, $\left|\lambda_k^{(\omega)}\right| < 1$.

Definition:

Eigenvectors $\mathbf{v}^k$ with $1 \leq k < N/2$ are called smooth (low-frequency).

Those with $N/2 \leq k < N$ are called rough or oscillatory (high-frequency).

Recall that $\lambda_k = \cos\left(\frac{k\pi}{N}\right)$, so for rough eigenvectors,

$$\lambda_k \leq 0.$$

Exercise: Find $0 < \omega < 1$ which yields optimal convergence for the set of rough modes for arbitrary N:

$$\omega: \quad \sup_{N} \max_{\frac{N}{2} \le k < N} \left| \lambda_k^{(\omega)} \right| = \min!,$$

i.e.,

$$\omega: \quad \sup_{\lambda \in (-1,0]} \left| 1 - \omega + \omega\lambda \right| = \min!,$$

What is then the bound on the convergence factor, $\left| \lambda_k^{(\omega)} \right|$, maximized over the rough modes? (Clues in my introductory paper.)

1D Model Problem

Find u which satisfies:

$$Lu = u''(x) = f(x) \ , \ \ x \in (0, \ 1) \ , \qquad (1)$$

$$u(0) = u_0 ,$$

$$u(1) = u_1 .$$

(In the particular case where $f = 0$, the solution is a straight line that connects u_0 with u_1.)

Discrete approximation: Since closed-form solutions exist only for a small number of differential equations, we solve such equations approximately by a discrete approximation.

Define a grid: divide the domain $(0,1)$ into N intervals. Assume for simplicity a uniform grid of mesh-size $h=1/N$.

Finite-difference discretization; examples:

Forward differences:

$$u' = \frac{u(x+h)-u(x)}{h} + O(h).$$

Backward differences:

$$u' = \frac{u(x)-u(x-h)}{h} + O(h).$$

Central differences:

$$u' = \frac{u(x+h)-u(x-h)}{2h} + O(h^2).$$

Second derivative:

$$u''(x) = \frac{u(x-h)-2u(x)+u(x+h)}{h^2} + O(h^2). \qquad (2)$$

Derivation: by the Taylor theorem

We can thus approximate the differential equation by a set of algebraic difference equations:

$$L^h u^h = \frac{u_{i+1}^h - 2u_i^h + u_{i-1}^h}{h^2} = f_i^h,$$

$$i = 1, \ldots, N-1,$$

$$u_0^h = u_0,$$

$$u_N^h = u_1.$$

In matrix form:

$$\frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \cdots & \cdots & \cdots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix} \begin{bmatrix} u_1^h \\ u_2^h \\ \cdots \\ u_{N-2}^h \\ u_{N-1}^h \end{bmatrix} =$$

$$\begin{bmatrix} f_1^h - u_0^h / h^2 \\ f_2^h \\ \cdots \\ f_{N-2}^h \\ f_{N-1}^h - u_1^h / h^2 \end{bmatrix} .$$

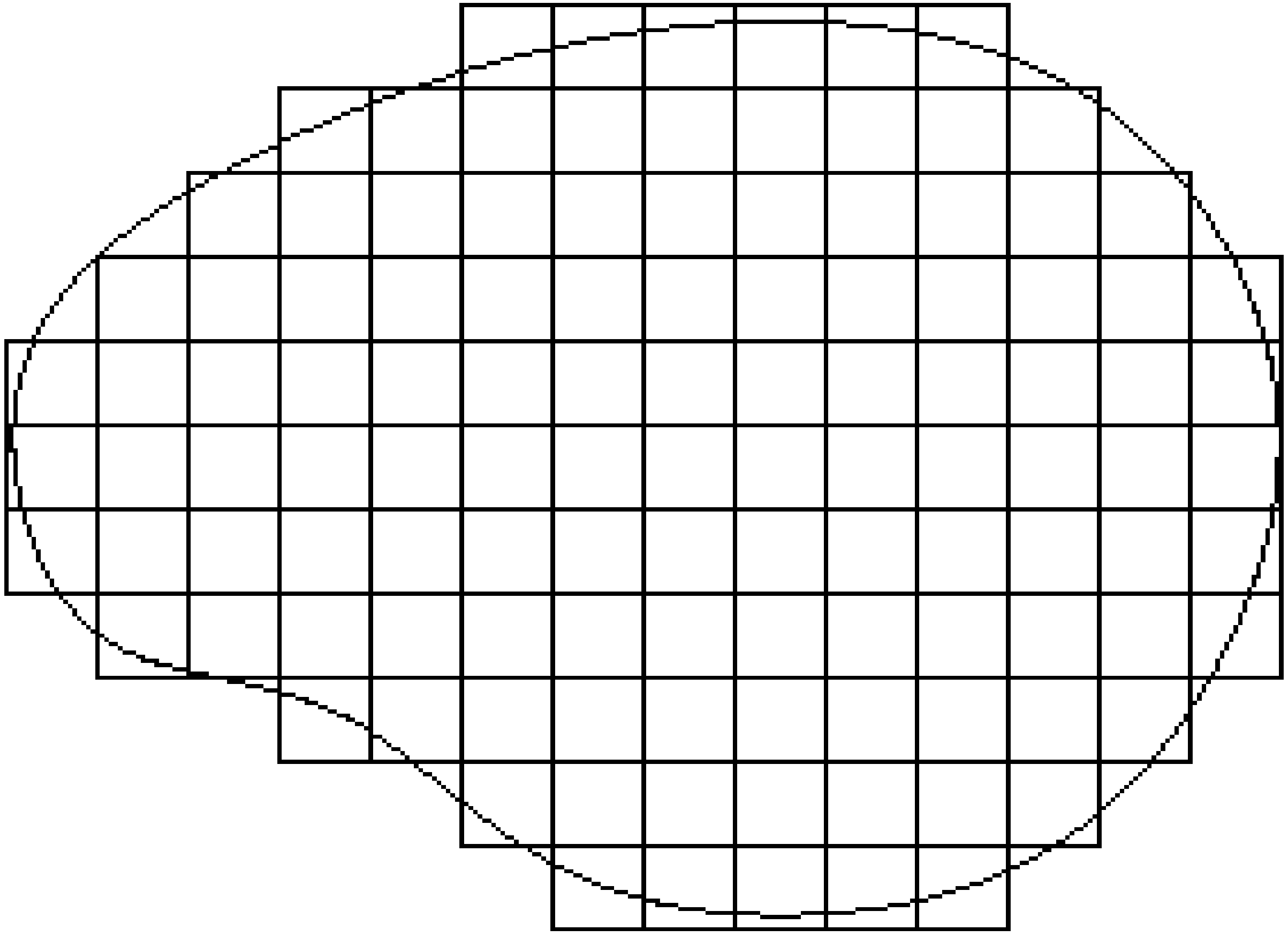
This is a tridiagonal system of equations which can be solved directly or iteratively.

2D Model Problem

Find u which satisfies:

$$\begin{aligned} Lu = u_{xx} + u_{yy} &= f(x,y), \quad (x,y) \in \Omega, \\ u &= g(x,y), \quad (x,y) \in \partial\Omega. \end{aligned} \tag{4}$$

This is the 2D Poisson equation, with Dirichlet boundary conditions. It is an elliptic partial differential equation which appears in many models.



Ω^h

Discrete approximation

Define a grid: $\Omega^h \subset \Omega$ (assumed to be uniform for simplicity, with mesh interval h).

Let u^h, g^h and f^h denote discrete approximations to u, g and f defined at the nodes of the grid.

Plug (2) for u_{xx}, and the analogous approximation for u_{yy} into (4), obtaining:

$$L^h u^h_{i,j} = \tag{5}$$

$$\frac{u^h_{i-1,j} - 2u^h_{i,j} + u^h_{i+1,j}}{h^2} + \frac{u^h_{i,j-1} - 2u^h_{i,j} + u^h_{i,j+1}}{h^2} = f^h_{i,j} \text{ in } \Omega^h$$

$$u^h = g^h \text{ on } \partial^h \Omega^h$$

This yields a nonsingular linear system of equations for $u^h_{i,j}$

We consider solving this system by the classical approach of Gauss-Seidel relaxation.

Gauss-Seidel (GS) Relaxation:

1. Choose initial guess, $\tilde{u}^h$.
2. Repeat until some convergence criterion is satisfied

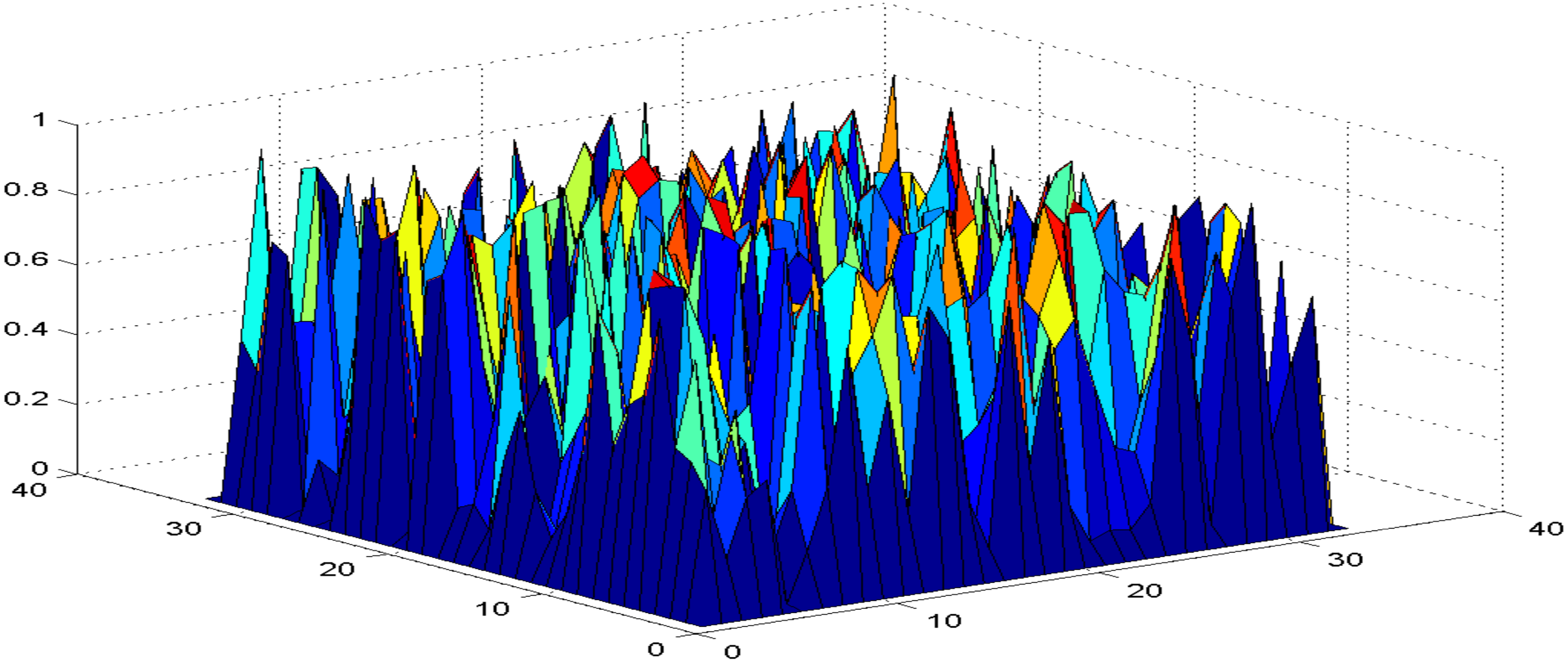
   {

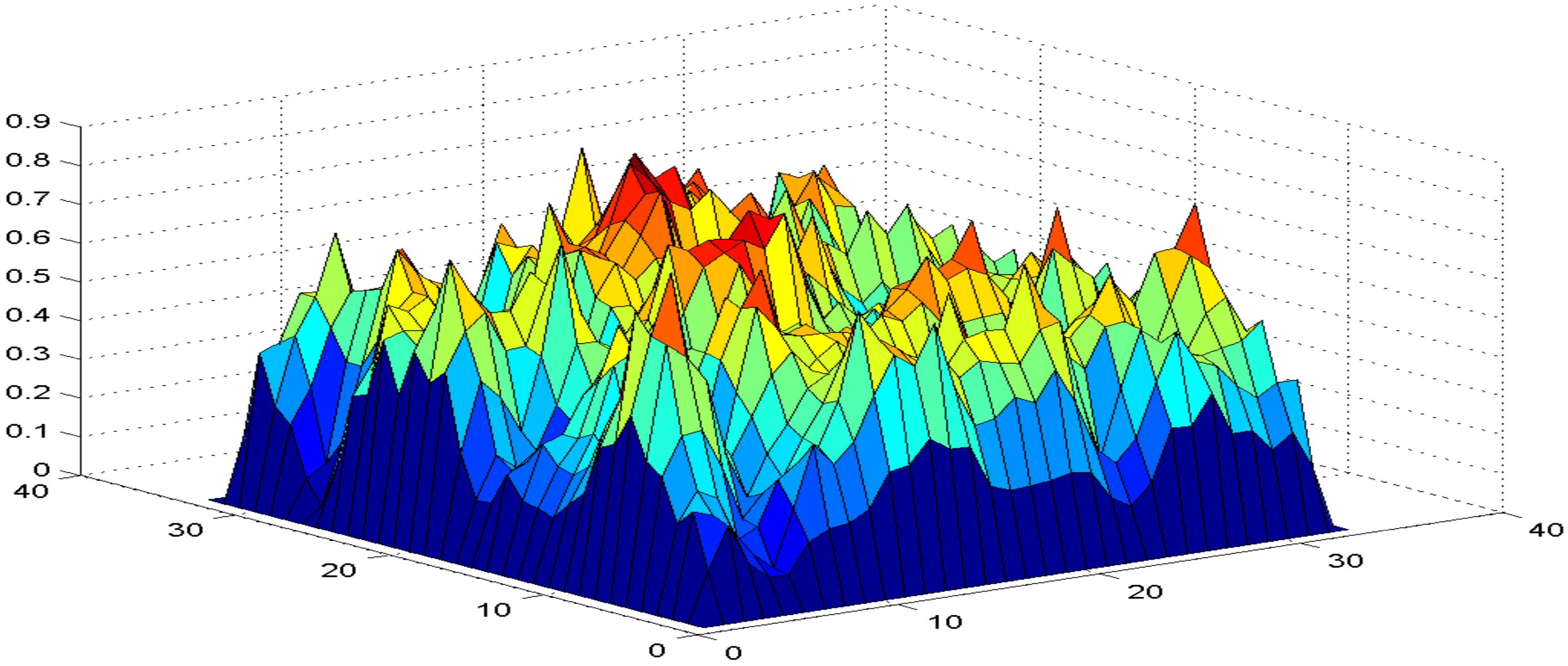
   Scan all variables in some prescribed order, and change each variable $\tilde{u}^h_{i,j}$ in turn so as to satisfy the (i,j)th equation.

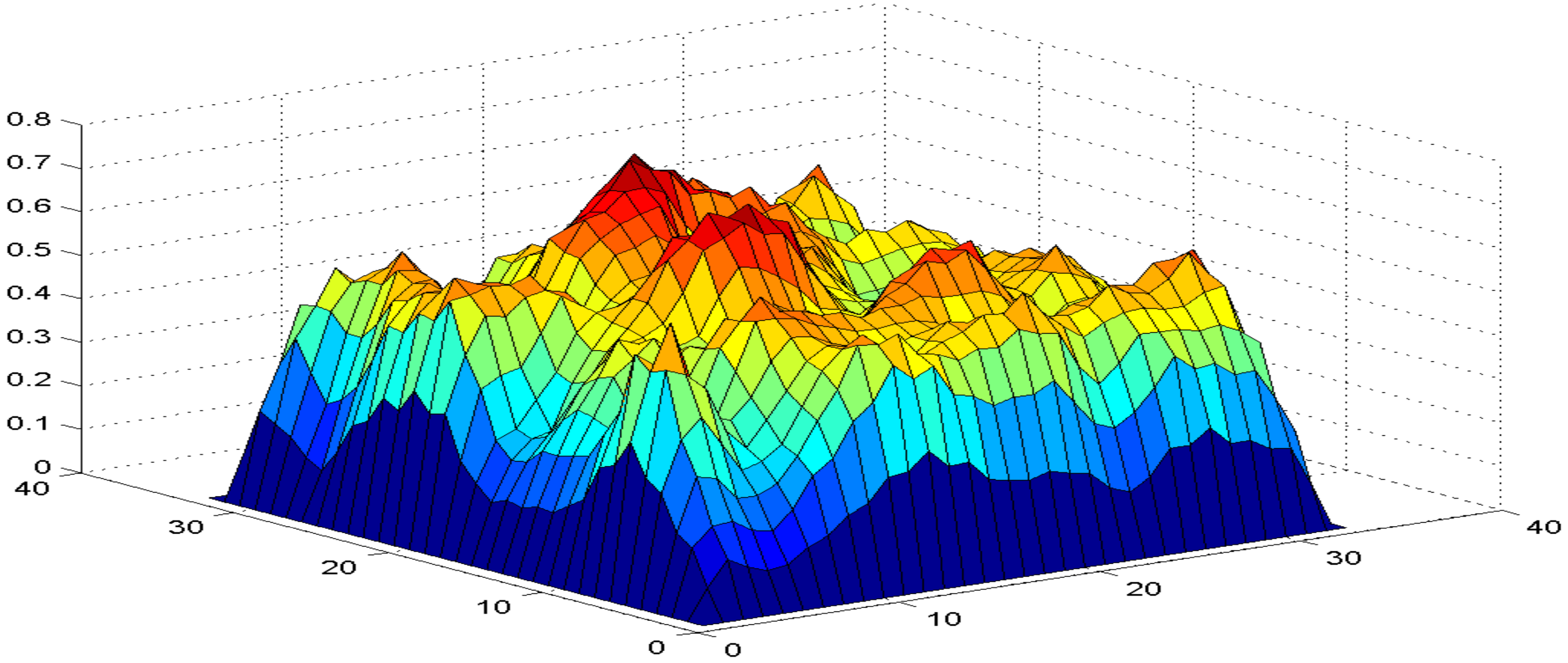
   }

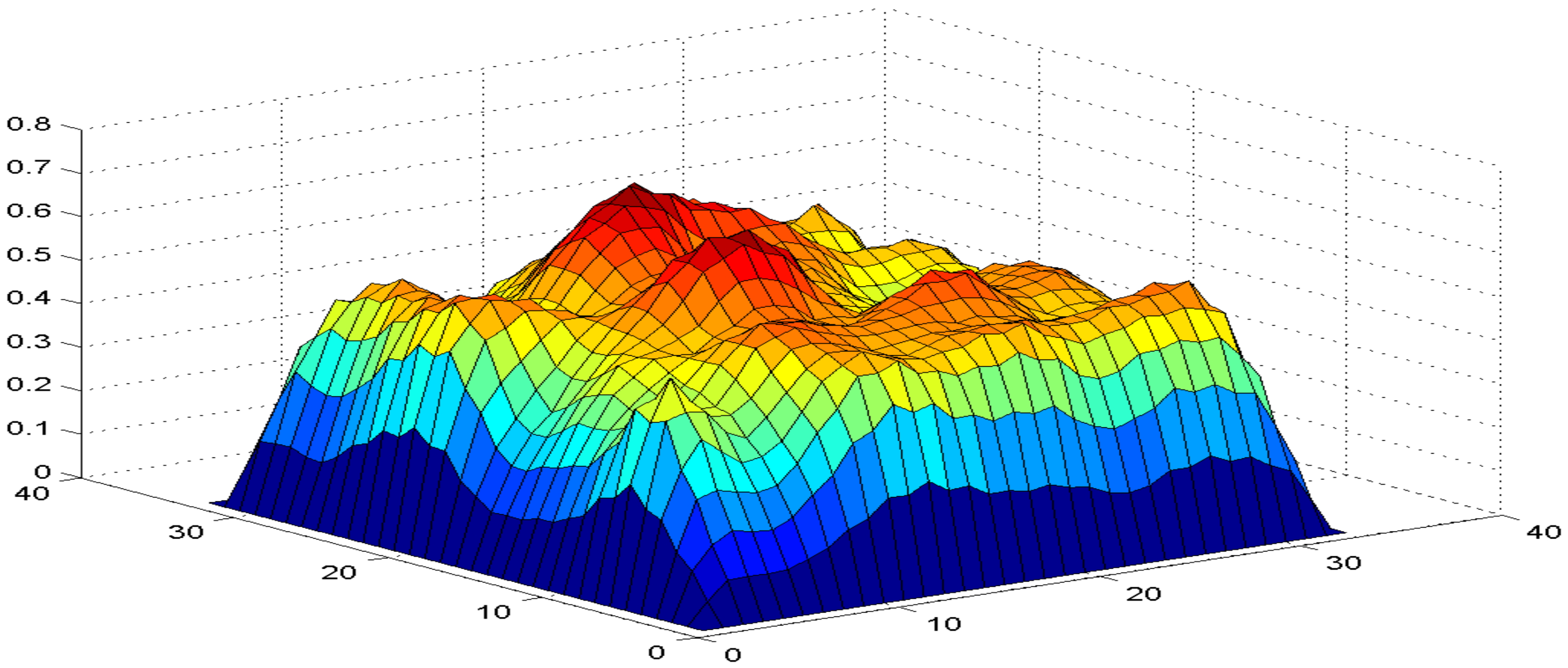
Observation: GS is a local process, because only near neighbors appear in each equation. Hence, it may be efficient for eliminating errors which can be detected locally. But large-scale ("smooth") errors are eliminated very slowly.

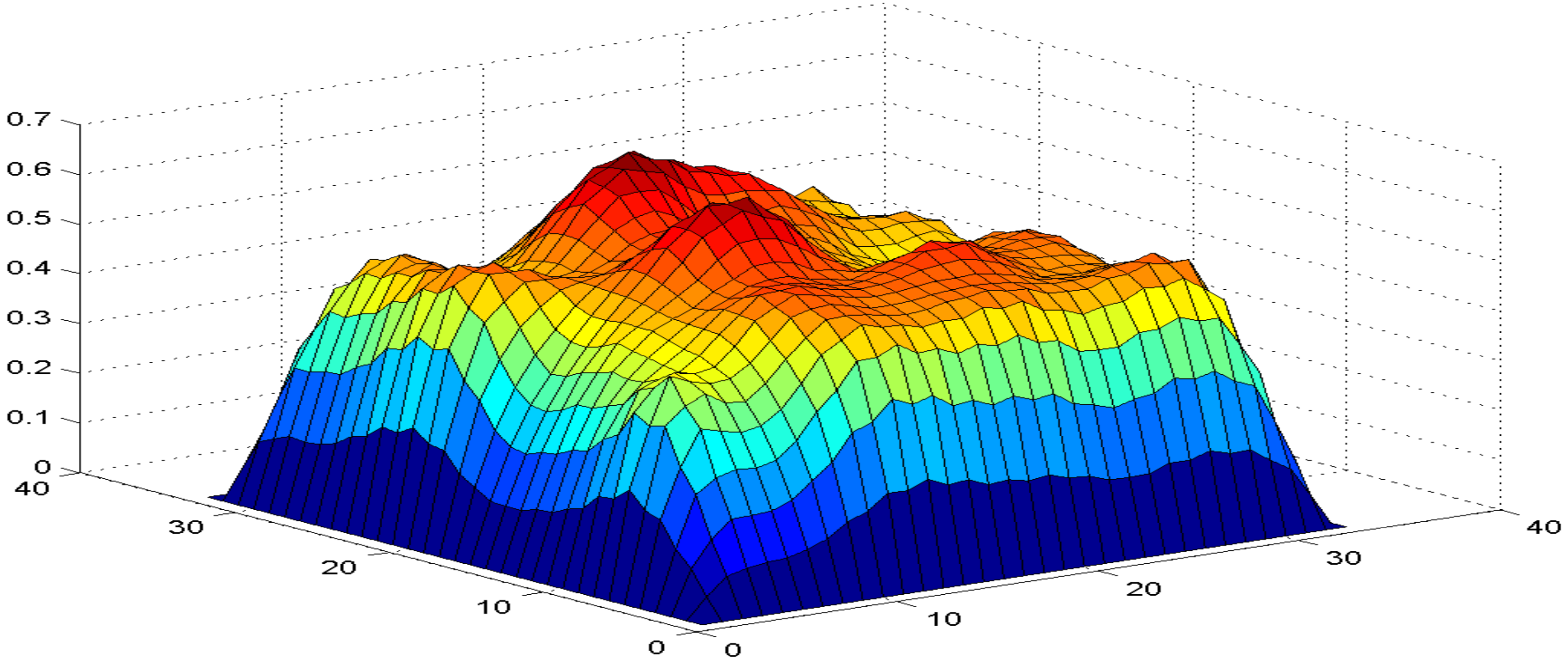
(The difference between GS and Jacobi is that old neighboring values are used in Jacobi, while the most updated values are used in GS.)

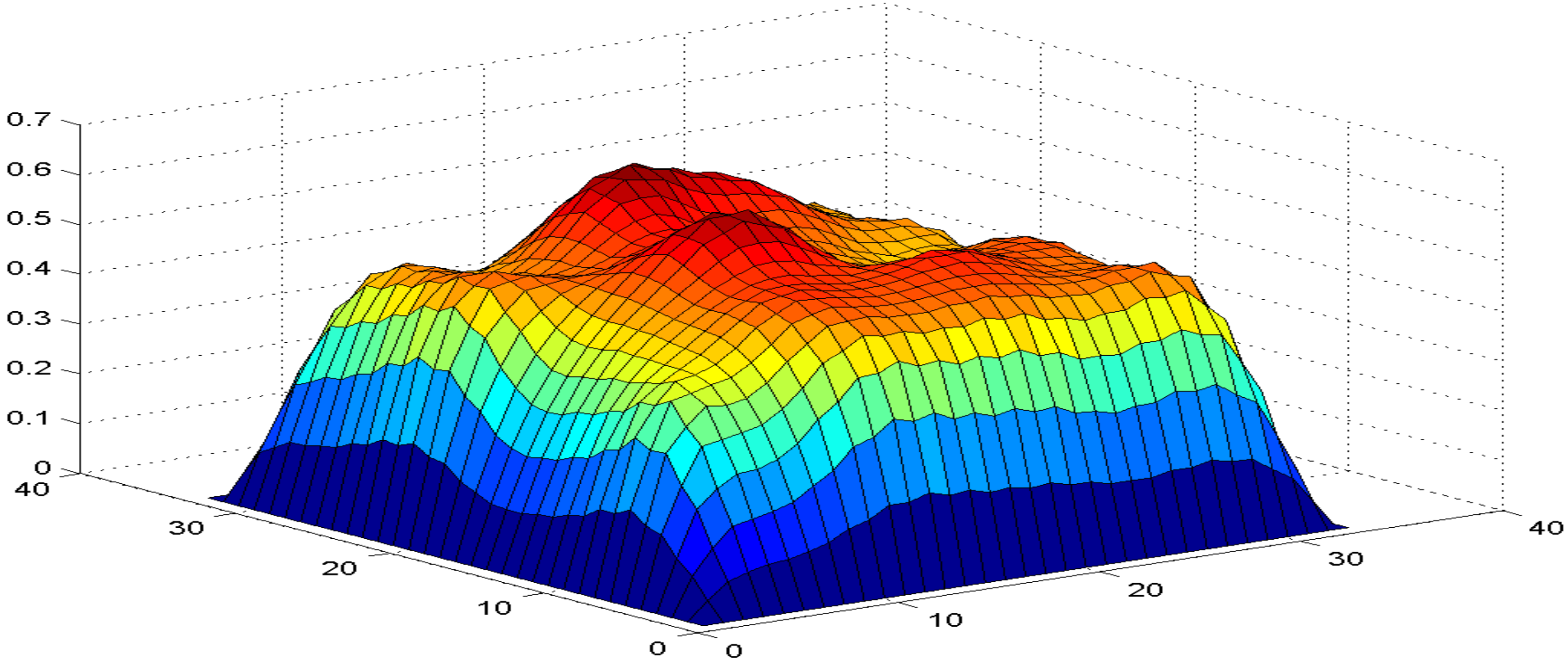

1
0.8
0.6
0.4
0.2
0
40
30
20
10
0
0
10
20
30
40

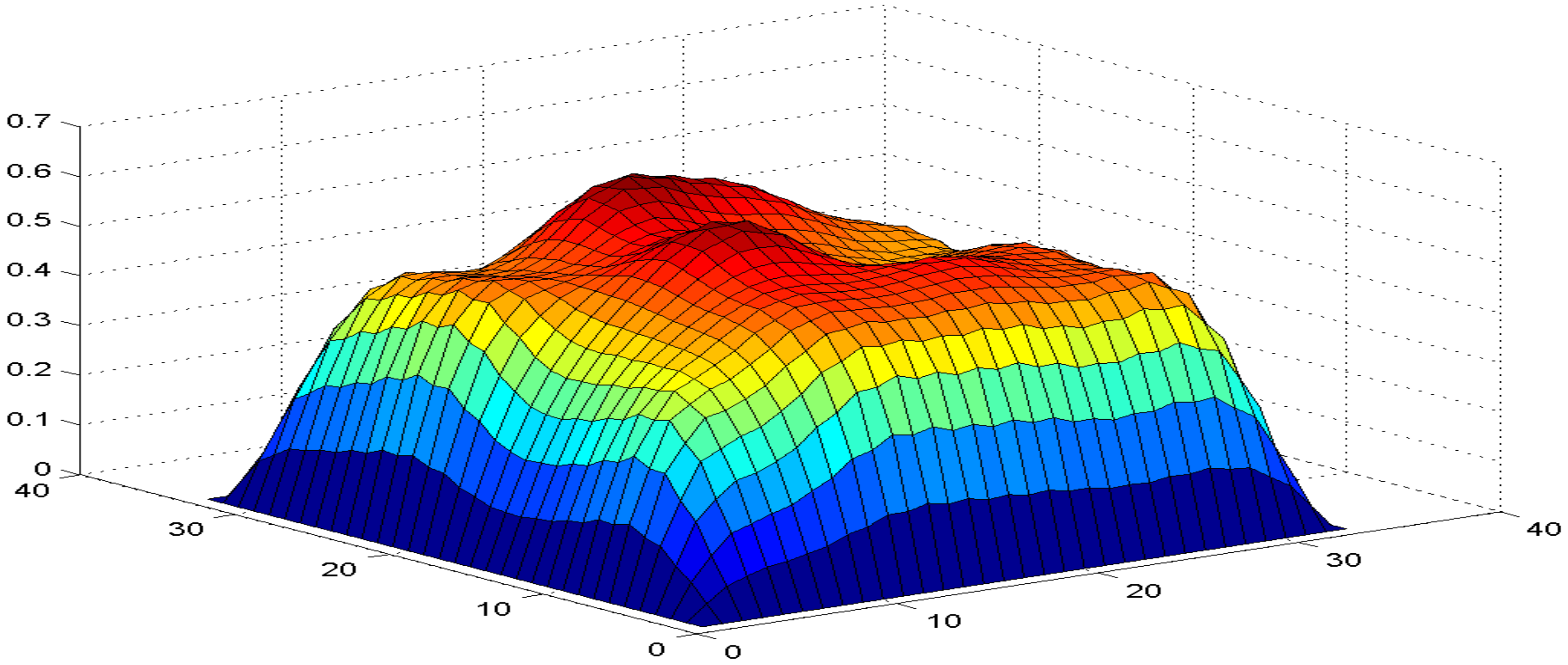

0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

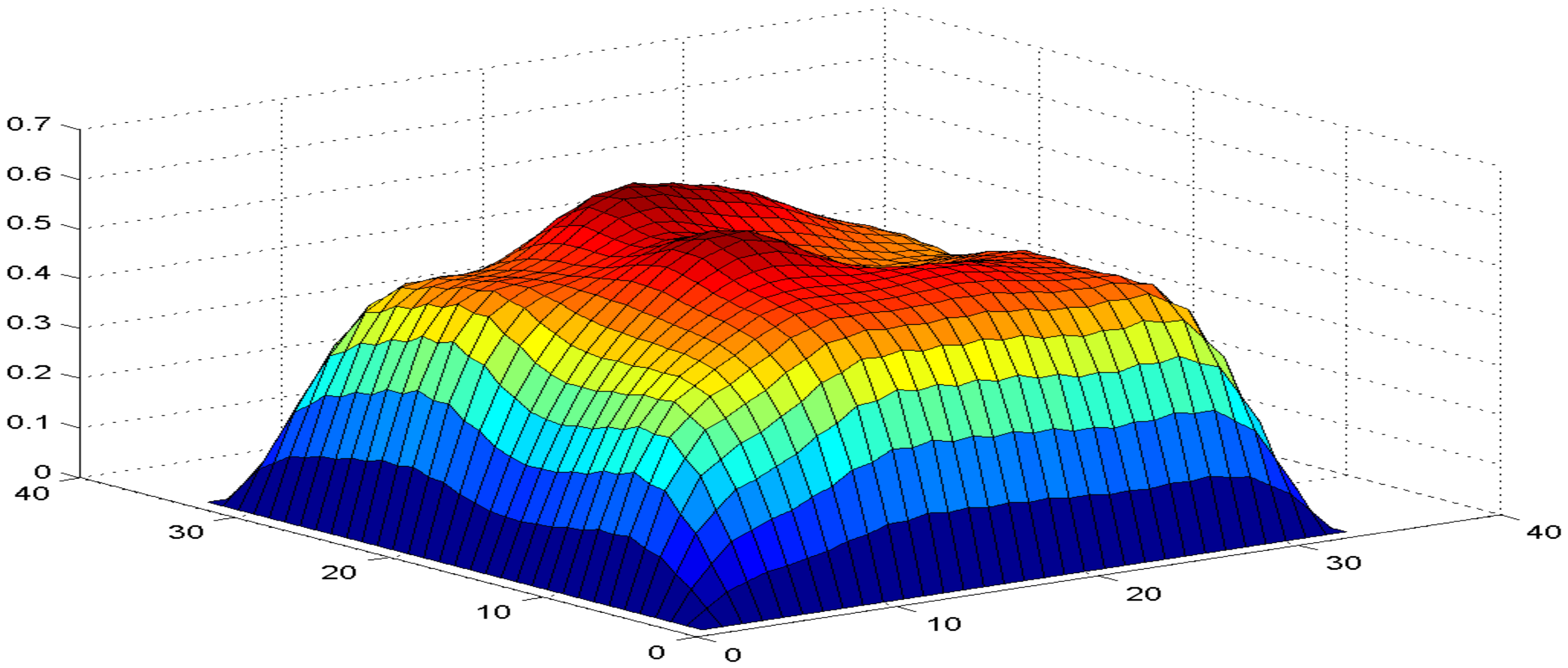

0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

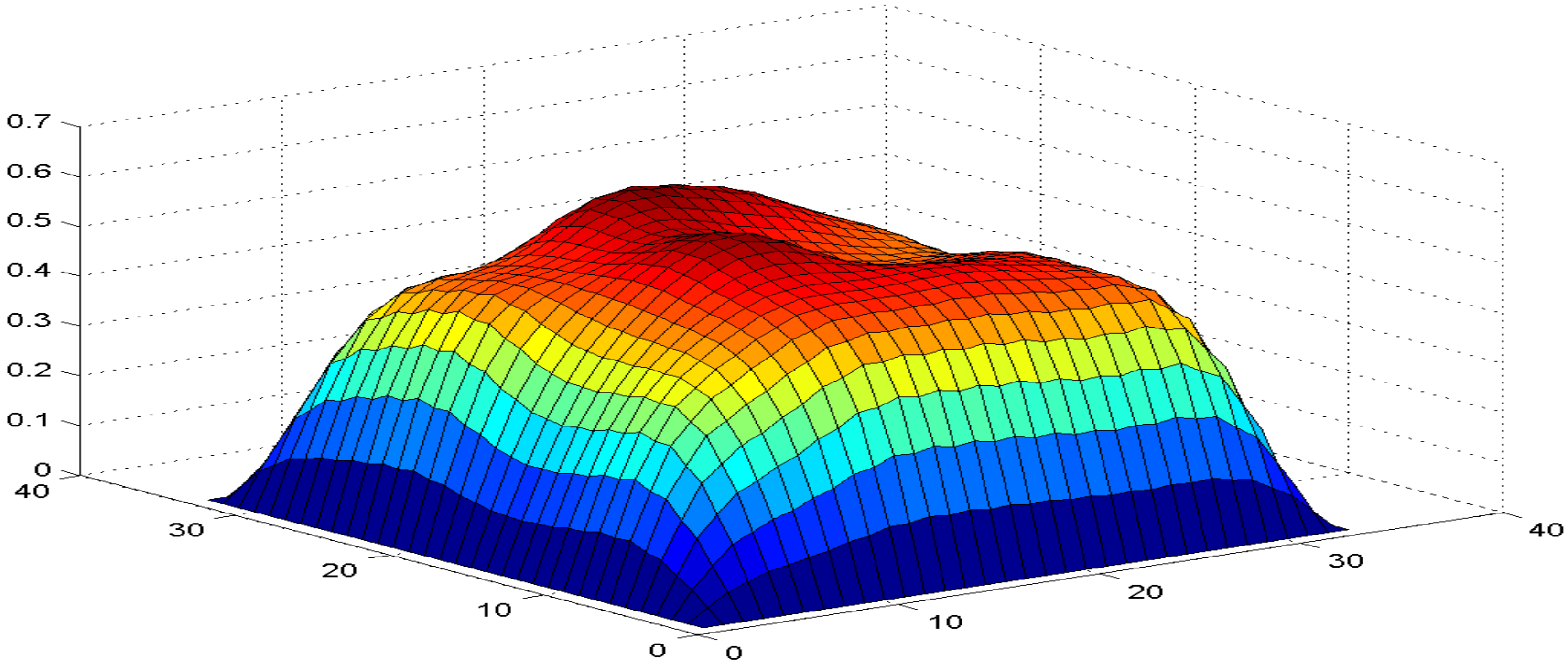

0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

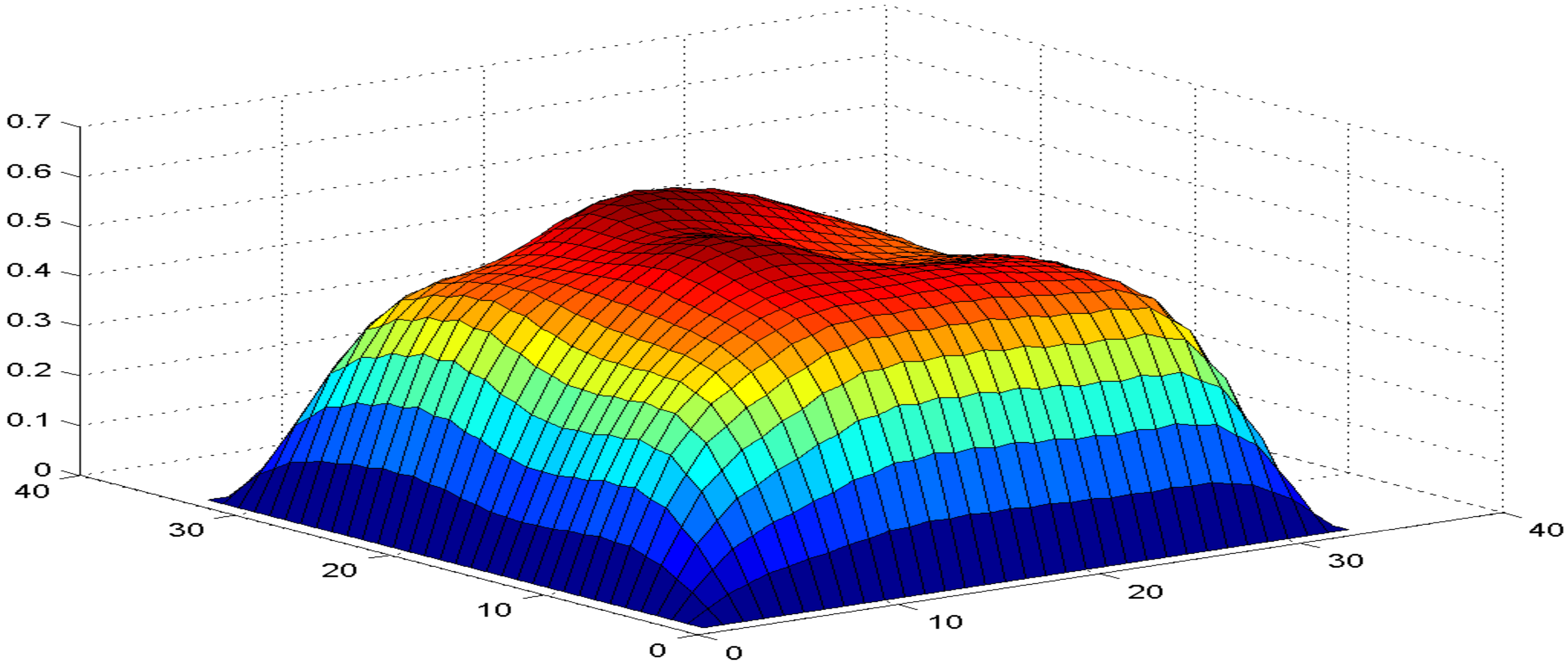

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

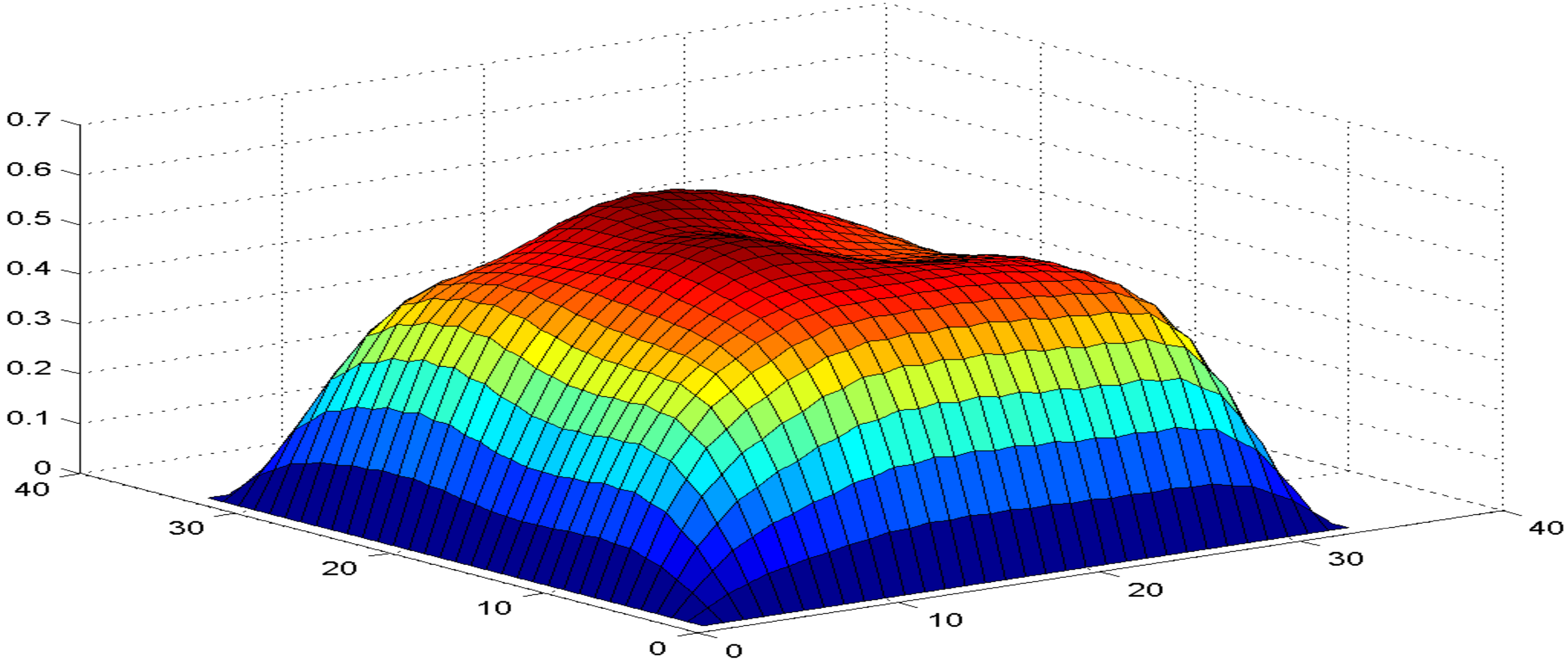

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

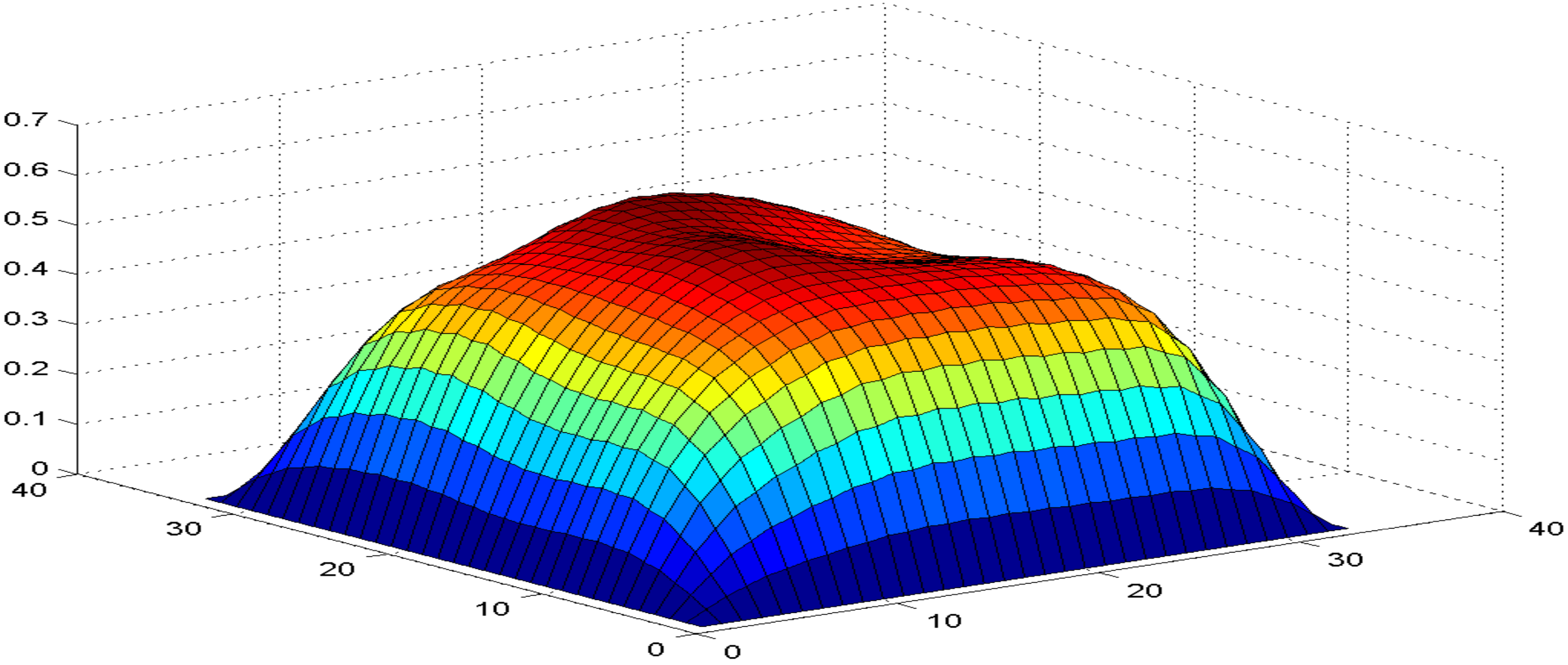

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

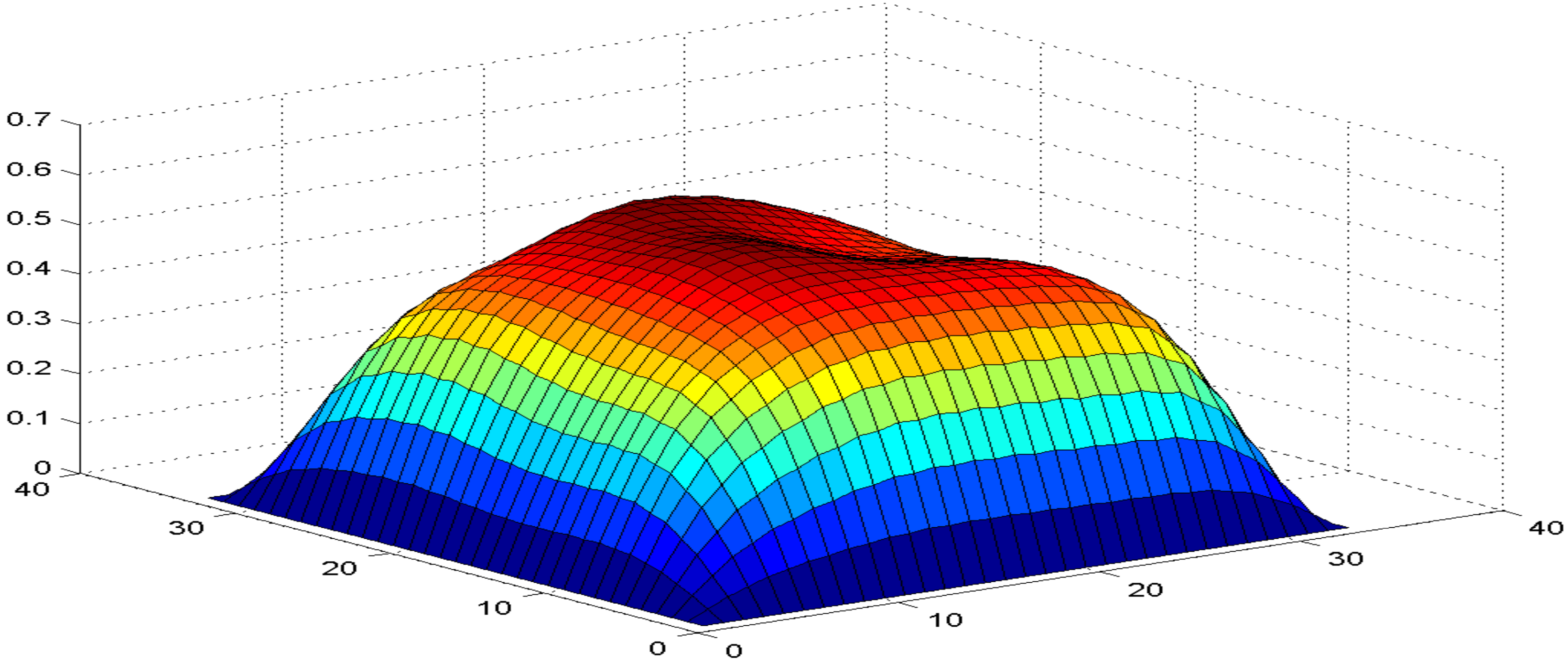

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

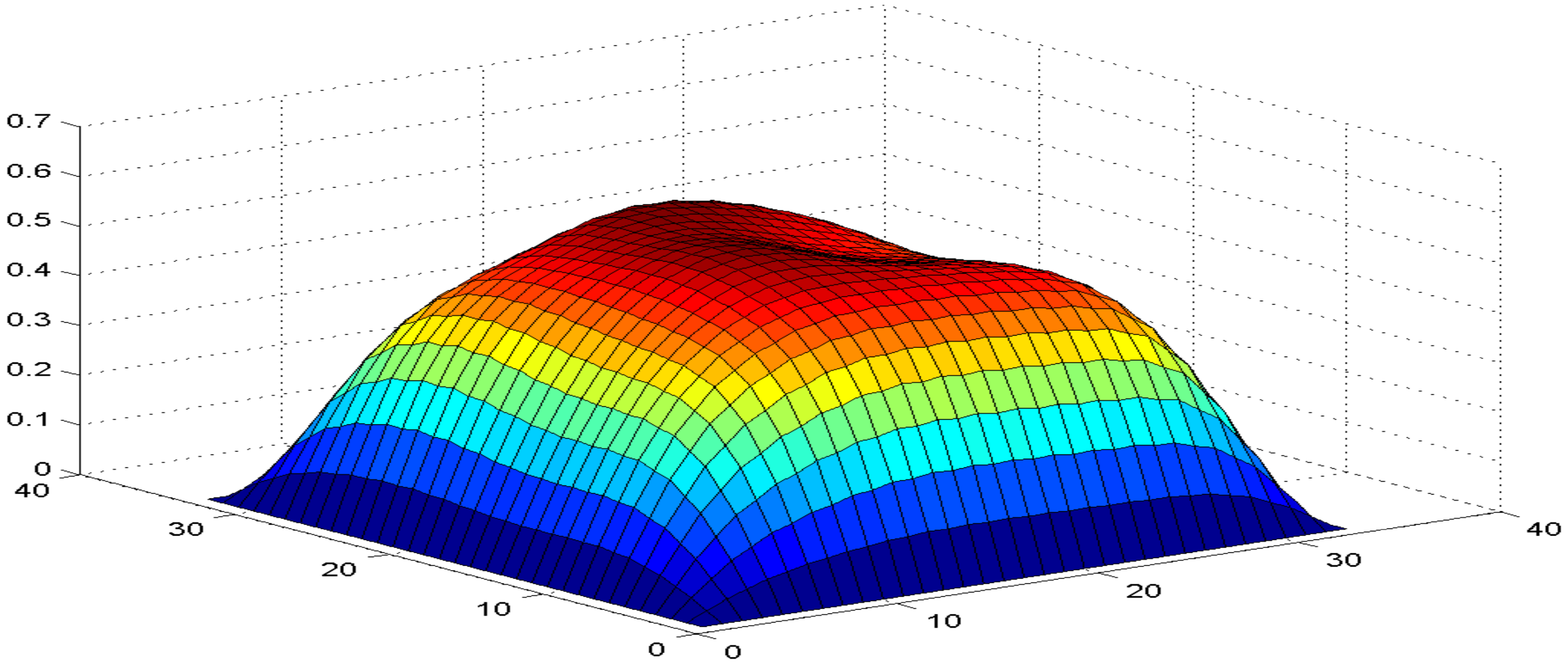

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

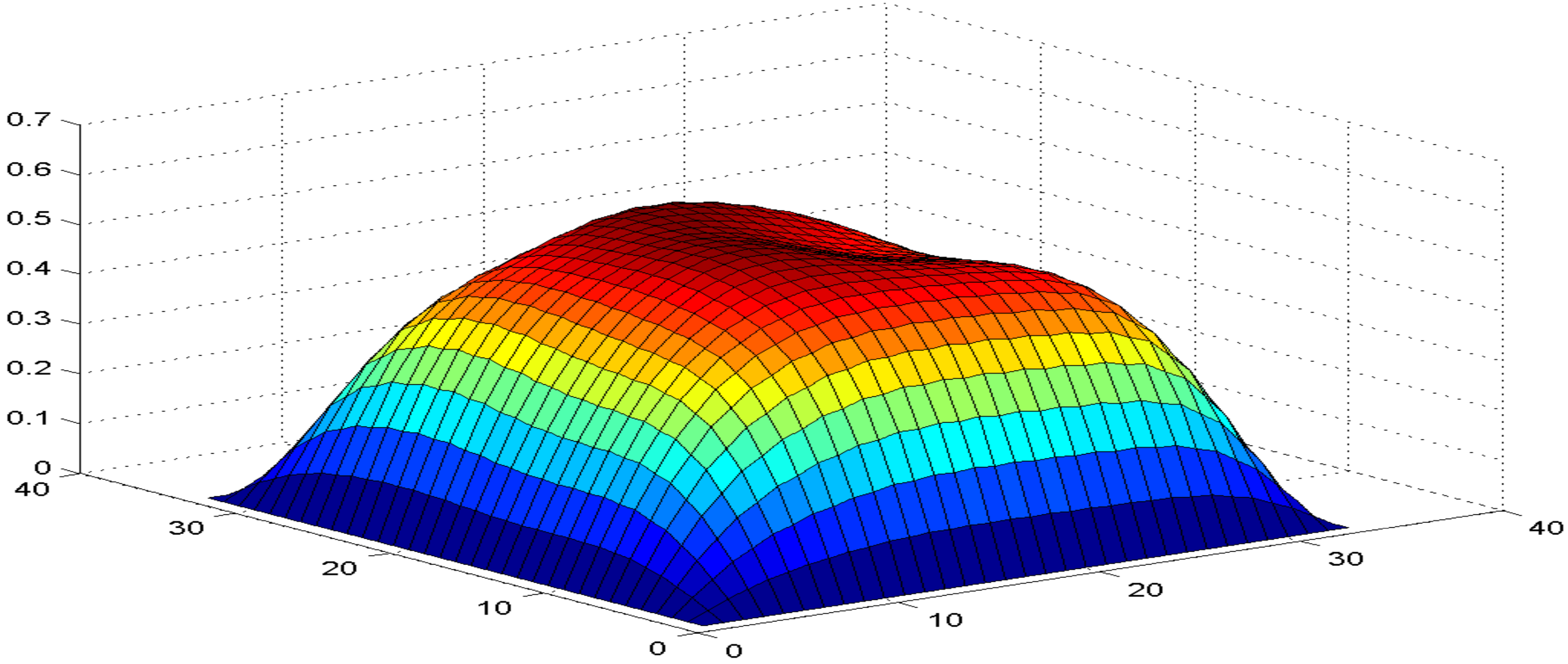

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

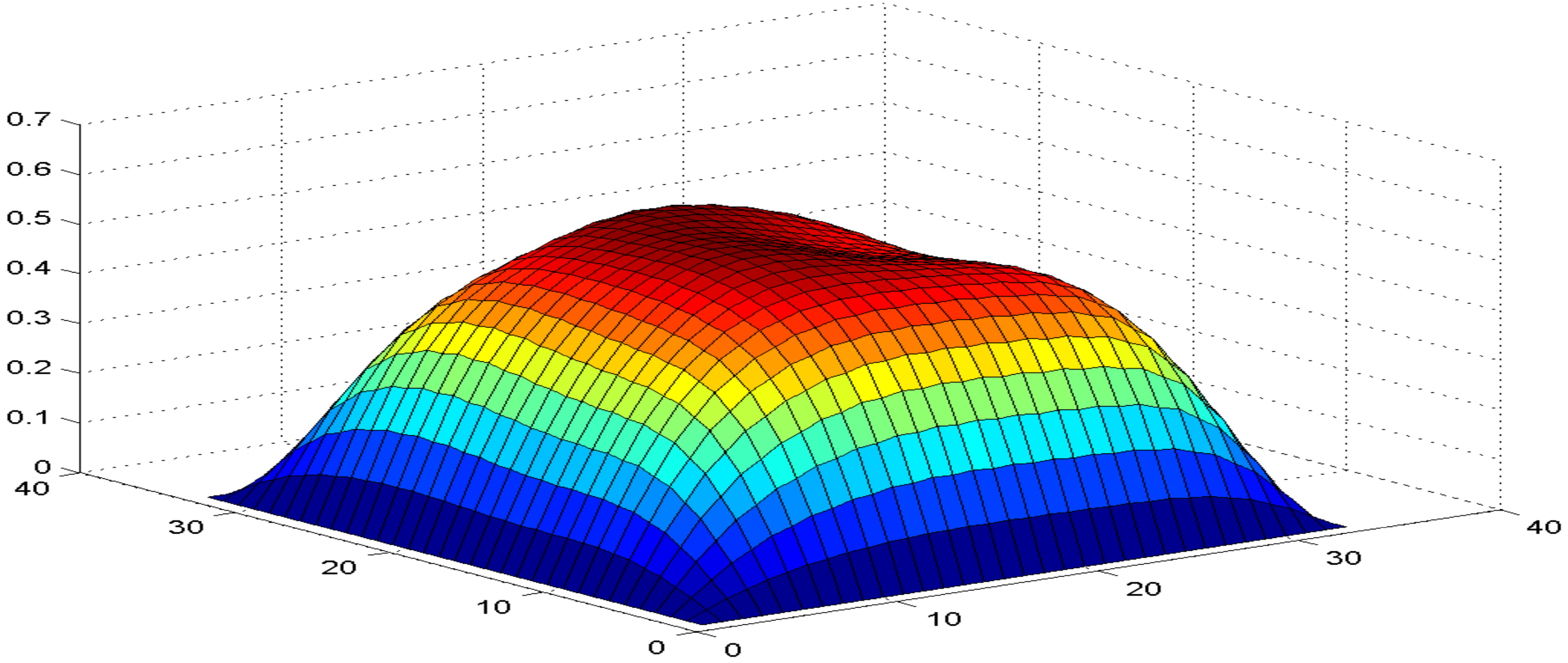

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

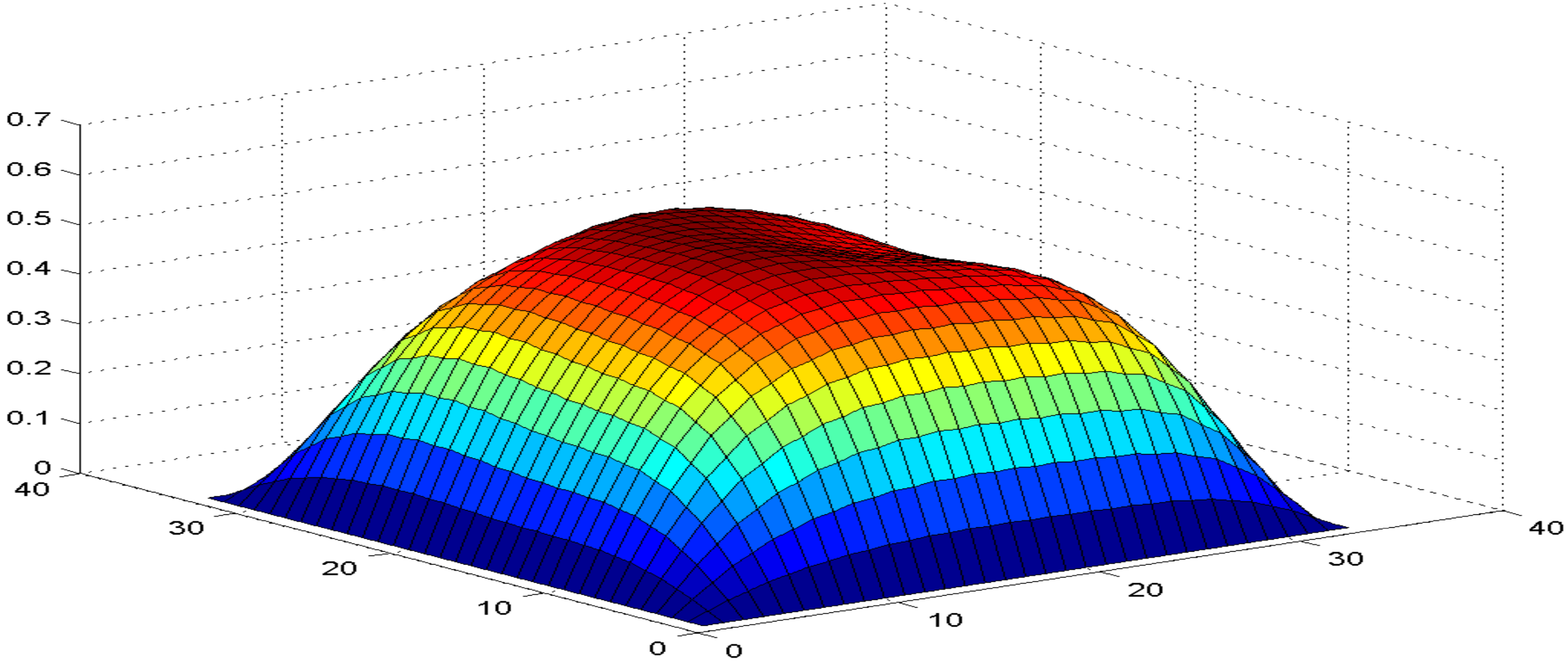

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

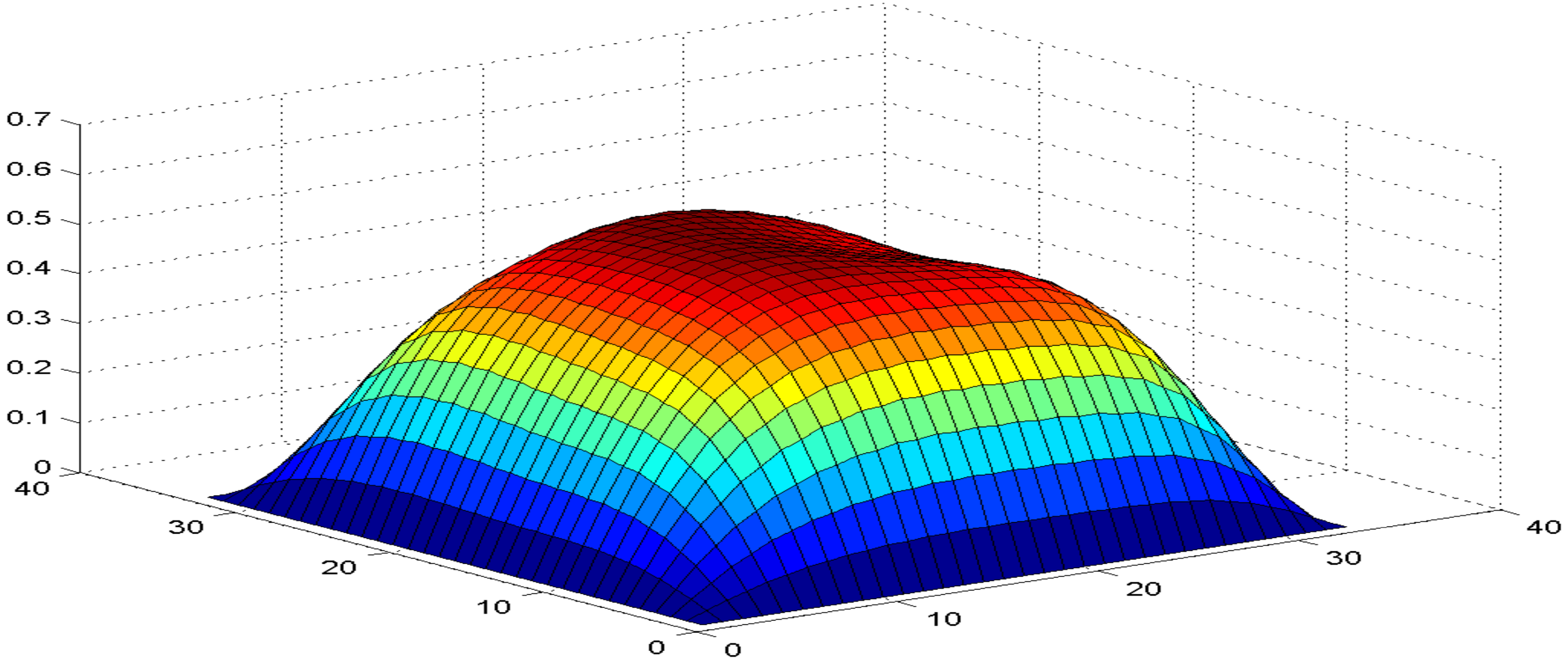

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

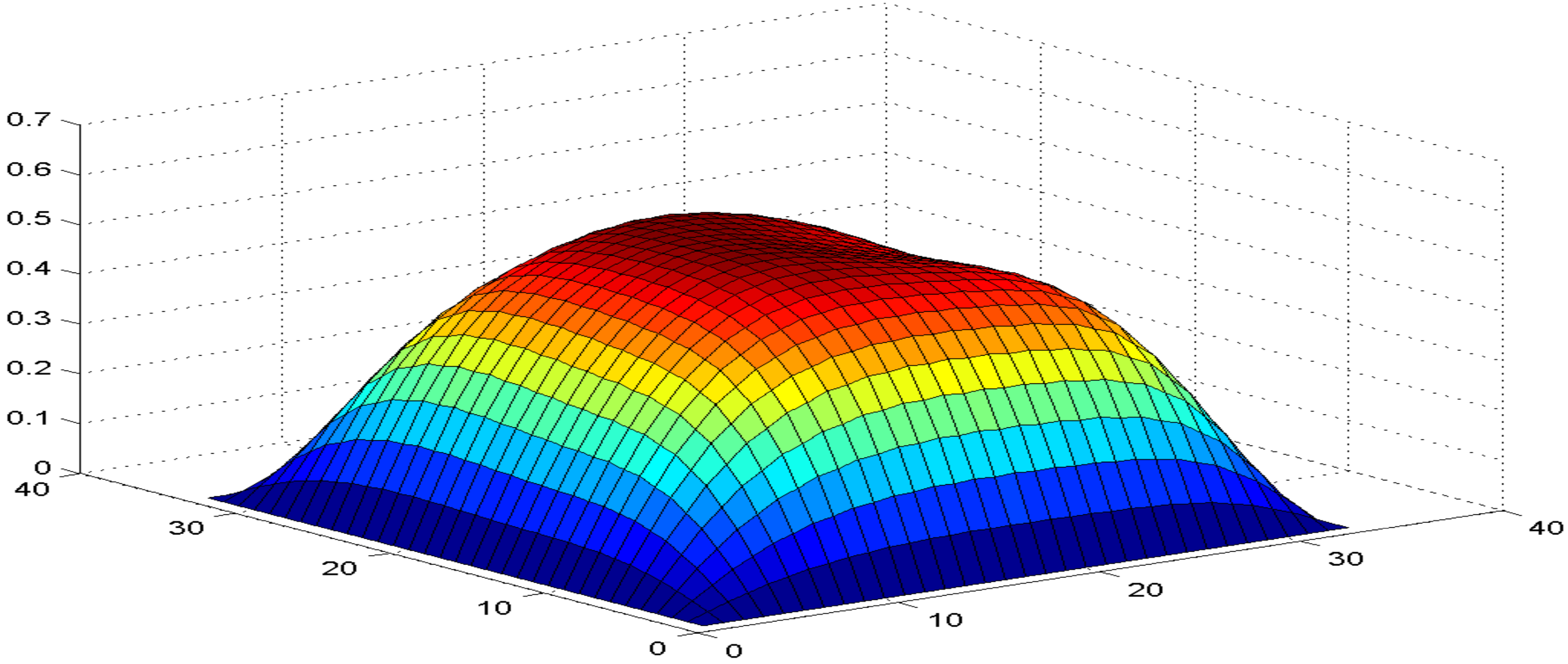

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

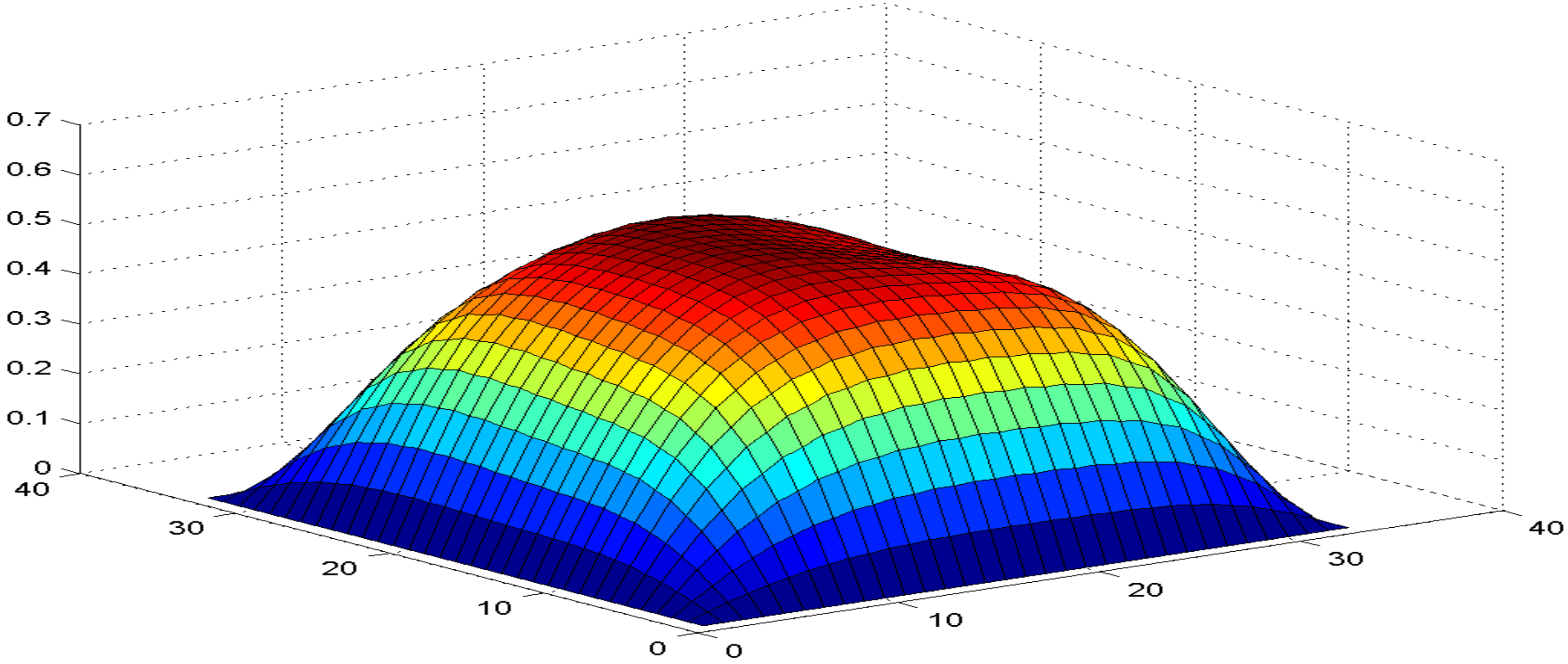

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

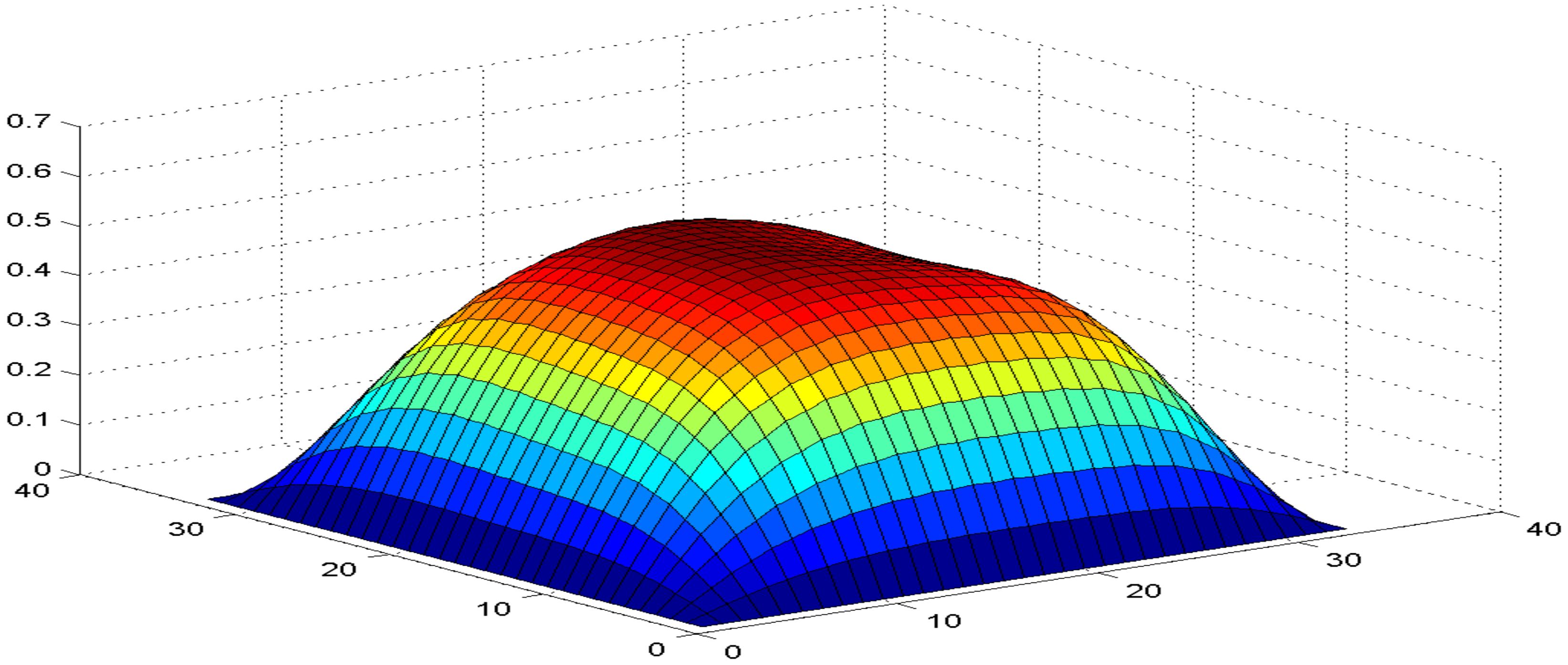

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

Key Observation re-worded: Relaxation cannot be generally efficient for reducing the error (i.e., the difference vector $\tilde{u}^h - u^h$). But relaxation may be extremely efficient for smoothing the error relative to the grid.

Practical conclusion:

1. A smooth error can be approximated well on a coarser grid.
2. A coarser grid implies less variables, hence less computation.
3. On the coarser grid the error is no longer as smooth relative to the grid, so relaxation may once again be efficient.

Grid-refinement algorithm

Define a sequence of progressively finer grids all covering the full domain. Then,

1. Define and solve the problem on the coarsest grid, say by relaxation.
2. Interpolate the solution to the next-finer grid. Apply several iterations of relaxation.
3. Interpolate the solution to the next-finer grid and continue in the same manner…

Does this method converge fast?

1D Model Problem Revisited

Fine-grid (h) difference equation:

$$L^h u^h = \frac{u_{i+1}^h - 2u_i^h + u_{i-1}^h}{h^2} = f_i^h, \qquad (6)$$

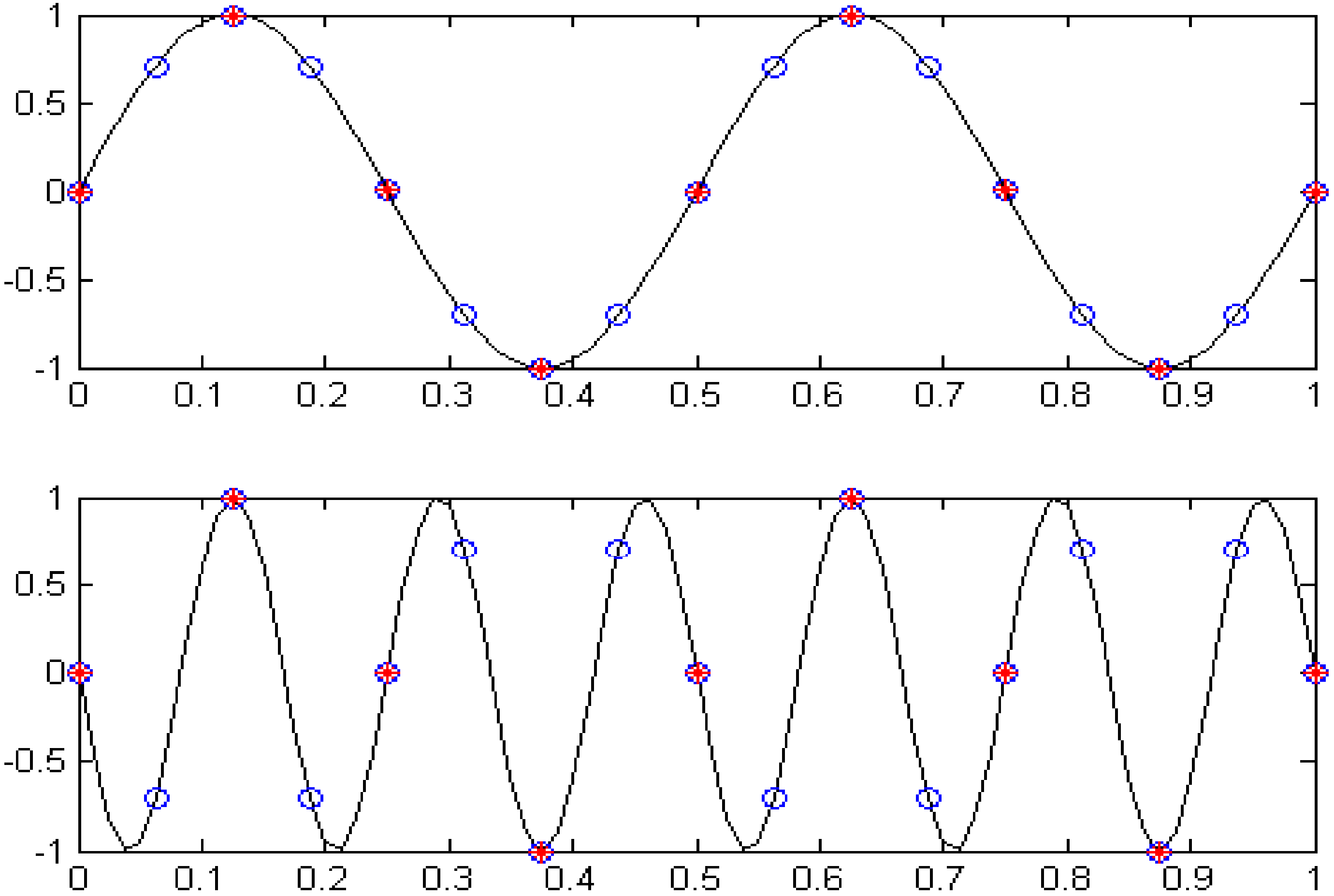
$$i = 1, \ldots N-1,$$

$$u_0^h = u_0,$$

$$u_N^h = u_1.$$

The eigenvectors of L^h (like those of the Jacobi relaxation operation) are Sine-function "waves":

$$\mathbf{v}^k = (\sin k\pi / N, \ldots \sin jk\pi / n, \ldots \sin(N-1)k\pi / N)^T \qquad (7)$$


1
0.5
0
-0.5
-1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
1
0.5
0
-0.5
-1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Aliasing

Smooth waves—with $k << N$—have wavelengths large compared to h. Hence they can be approximated well on the coarse grids. But non-smooth eigenvectors alias with smooth components on the coarse grids.

Since the right-hand side, f^h, will generally have some non-smooth components, these will be "interpreted" as smooth components by the coarse grids, resulting in a smooth error.

Hence, when we interpolate a coarse-grid solution to the fine grid, we still have smooth errors in this solution. These cannot be corrected efficiently by relaxation.

Errors:

There is an important distinction here between the discretization error:

$$u - u^h,$$

and the algebraic error:

$$u^h - \tilde{u}^h,$$

Where $\tilde{u}^h$ is our current approximation to u^h.

Note: Neither the solution, u^h, nor the discretization error are smoothed by relaxation, only the algebraic error. Hence, we formulate our problem in terms of this error.

Denote $$v^h = u^h - \tilde{u}^h.$$

Recall $$L^h u^h = f^h.$$

Subtract $L^h \tilde{u}^h$ from both sides, and use the linearity of L^h to obtain:

$$L^h v^h = f^h - L^h \tilde{u}^h \equiv r^h \tag{8}$$

As we have seen, we need to smooth the error v^h on the fine grid first, and only then solve the coarse-grid problem. Hence, we need two types of integrid transfer operations:

1. A Restriction (fine-to-coarse) operator: I_h^H.
2. A Prolongation (coarse-to-fine) operator: I_H^h.

Two-grid Algorithm

- Relax several times on grid h, obtaining $\widetilde{u}^h$ with a smooth corresponding error.
- Calculate the residual: $r^h = f^h - L^h\widetilde{u}^h.$
- Solve approximate error-equation on the coarse grid:

$$L^H v^H = f^H \equiv I_h^H r^h.$$

- Interpolate and add correction:

$$\widetilde{u}^h \leftarrow \widetilde{u}^h + I_H^h v^H.$$

- Relax again on grid h.

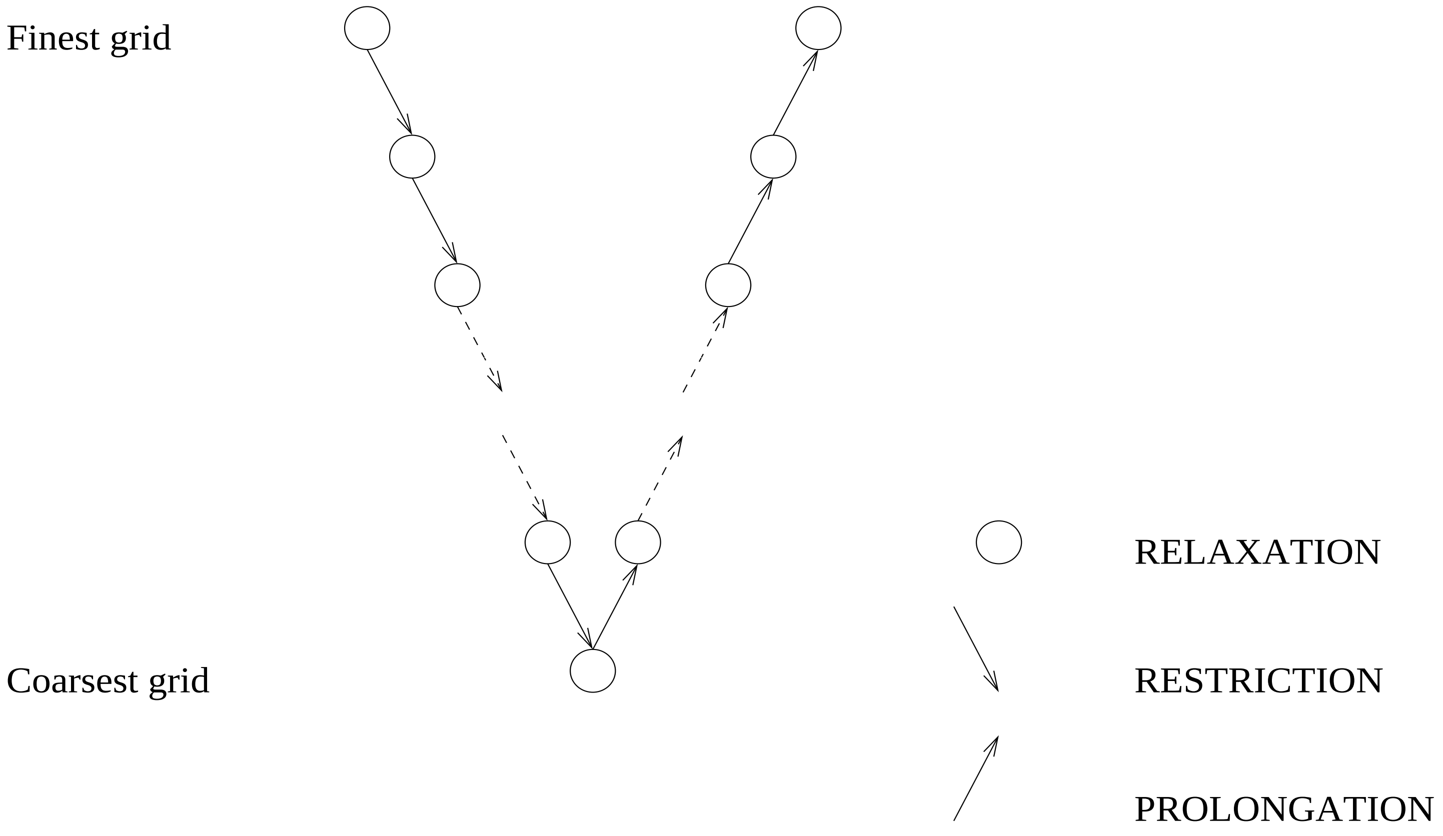
Multi-grid is obtained by recursion.

Multi-grid Cycle $V(\nu_1,\nu_2)$

Let u^{2h} approximate v^{2h}, u^{4h} approximate the error on grid $2h$, etc.

```
Relax on $L^h u^h = f^h$ $\nu_1$ times
Set $f^{2h} = I_h^{2h}(f^h - L^h u^h)$, $u^{2h} = 0$
    Relax on $L^{2h} u^{2h} = f^{2h}$ $\nu_1$ times
    Set $f^{4h} = I_{2h}^{4h}(f^{2h} - L^{2h} u^{2h})$, $u^{4h} = 0$
        Relax on $L^{4h} u^{4h} = f^{4h}$ $\nu_1$ times
        Set $f^{8h} = I_{4h}^{8h}(f^{4h} - L^{4h} u^{4h})$, $u^{8h} = 0$
            $\dots$
            Solve $L^{Mh} - u^{Mh} = f^{Mh}$
            $\dots$
        Correct $u^{4h} \leftarrow u^{4h} + I_{8h}^{4h} u^{8h}$
        Relax on $L^{4h} u^{4h} = f^{4h}$ $\nu_2$ times
    Correct $u^{2h} \leftarrow u^{2h} + I_{4h}^{2h} u^{4h}$
    Relax on $L^{2h} u^{2h} = f^{2h}$ $\nu_2$ times
Correct $u^h \leftarrow u^h + I_{2h}^h u^{2h}$
Relax on $L^h u^h = f^h$ $\nu_2$ times
```

V cycle



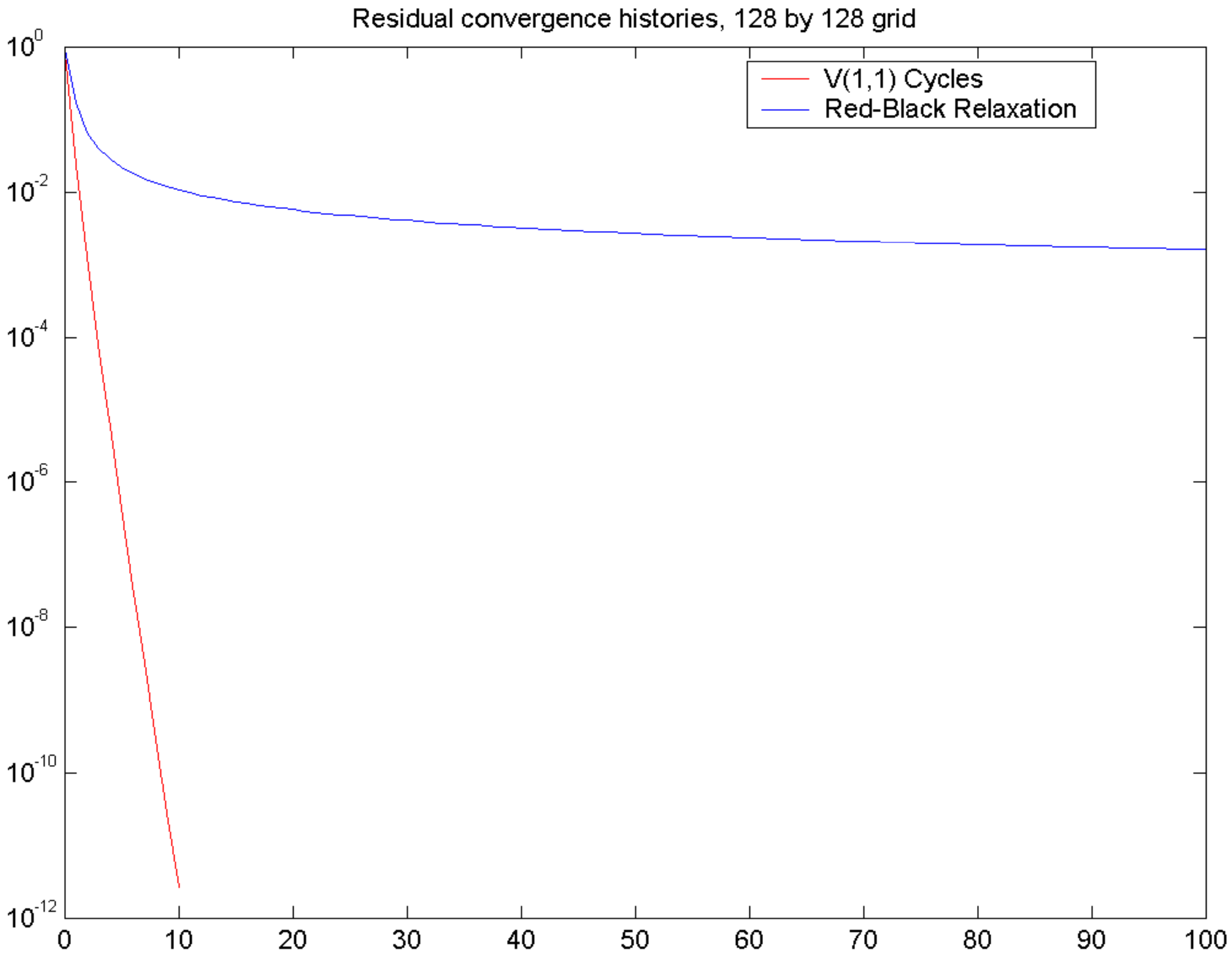
Finest grid

Coarsest grid

○ RELAXATION

↘ RESTRICTION

↗ PROLONGATION



Multigrid vs. Relaxation

Variational Coarsening

All of this is well and good when we have a straightforward structured problem derived from a partial differential equation.

How should we choose I_h^H, I_H^h, L^H in more general situations?

Variational Coarsening

Assume that L^h is symmetric positive definite (SPD).

Let us then recast our problem as a convex functional minimization task:

$$u^h = \arg\min_{v^h} \left(\frac{1}{2} \left(v^h\right)^T L^h v^h - \left(v^h\right)^T f^h \right)$$

$$\Leftrightarrow$$

$$L^h u^h - f^h = 0.$$

Variational Coarsening

Recall the coarse-grid correction step:

$$\tilde{u}^h \leftarrow \tilde{u}^h + I_H^h e^H .$$

Given $\tilde{u}^h$, our current approximation, we wish to add a correction $I_H^h e^H$ that will reduce the fine-grid functional as much as possible.

Note that the set of possible corrections is the space spanned by the columns of I_H^h, called the range of I_H^h.

Variational Coarsening

Plugging into the functional yields:

$$e^H = \arg\min_{c_H}$$

$$\left(\frac{1}{2}\left(\tilde{u}^h + I_H^h c^H\right)^T L^h \left(\tilde{u}^h + I_H^h c^H\right) - \left(\tilde{u}^h + I_H^h c^H\right)^T f^h \right)$$

$$\Leftrightarrow$$

$$\left(I_H^h\right)^T L_h I_H^h e^H + \left(I_H^h\right)^T \left(L^h \tilde{u}^h - f^h\right) = 0. \quad \left[L^H e^H = I_h^H r^h\right]$$

Choosing the Operators

Conclusions:

1. We should define $L_H = \left(I_H^h\right)^T L_h I_H^h$ (Galerkin coarsening) for the C.G. problem

$$L^H e^H = \left(I_H^h\right)^T r^h.$$

   Note that L^H is SPD.

2. We should choose I_H^h such that the unknown error, e^h, is approximately in its range.

This depends on the relaxation, hence on L^h.

Choosing the Operators

Related well-known observations:

1. If e^h is in the range of I_H^h prior to the coarse-grid correction, then it is eliminated exactly.

2. Amongst all possible coarse-grid corrections with the given I_H^h, the Galerkin correction minimizes the resulting fine-grid error in the energy norm:

$$\left\| e^h \right\|_{L^h}^2 = \left\langle \left(e^h \right)^T, L^h e^h \right\rangle = \left\langle \left(e^h \right)^T, r^h \right\rangle.$$

Nonlinear Multigrid Algorithm (FAS)

Recall that, in the usual multigrid approach, we use the coarse grid to approximate the correction to the fine-grid error. That is, we approximate the fine-grid equation

$$L^h v^h = r^h,$$

by the the coarse-grid equation

$$L^H v^H = I_h^H r^h.$$

We can rewrite the fine-grid equation as

$$L^h u^h - L^h \tilde{u}^h = r^h.$$

We approximate this equation on the coarse grid by

$$L^H u^H - L^H \tilde{u}^H = r^H,$$

with

$$\tilde{u}^H = I_h^H \tilde{u}^h.$$

The difference is that now the variable, $\tilde{u}^H$, approximates the full solution rather than just the correction. Hence, this approach can be applied to nonlinear problems. After we solve the coarse-grid problem, we interpolate and add the correction:

$$\tilde{u}^h \leftarrow \tilde{u}^h + I_H^h \left(u^H - \tilde{u}^H \right).$$

Two-grid FAS Algorithm

- Relax several times on grid h, obtaining $\tilde{u}^h$ with a smooth corresponding error.
- Calculate the residual: $r^h = f^h - L^h \tilde{u}^h$.
- Solve approximate equation for the full solution on the coarse grid: $L^H u^H = f^H \equiv I_h^H r^h + L^H \hat{I}_h^H \tilde{u}^h$.
- Interpolate and add correction:

$$\tilde{u}^h \leftarrow \tilde{u}^h + I_H^h \left(u^H - \hat{I}_h^H \tilde{u}^h \right).$$

- Relax again on grid h.

Multi-grid is obtained by recursion.

Conclusions

With proper care and insight, multilevel methods are a highly efficient tool for the iterative solution of problems such as those arising from the discretization of elliptic PDE, as well as many other types of problems.

Skill in developing efficient multilevel algorithms is required for:

1. Choosing a good local iteration.
2. Choosing appropriate coarse-scale variables.
3. Choosing inter-scale transfer operators.
4. Constructing coarse-scale approximations to the fine-scale problem.