

# A Multigrid Tutorial

Irad Yavneh

Department of Computer Science

Technion - Israel Institute of Technology

[irad@cs.technion.ac.il](mailto: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.

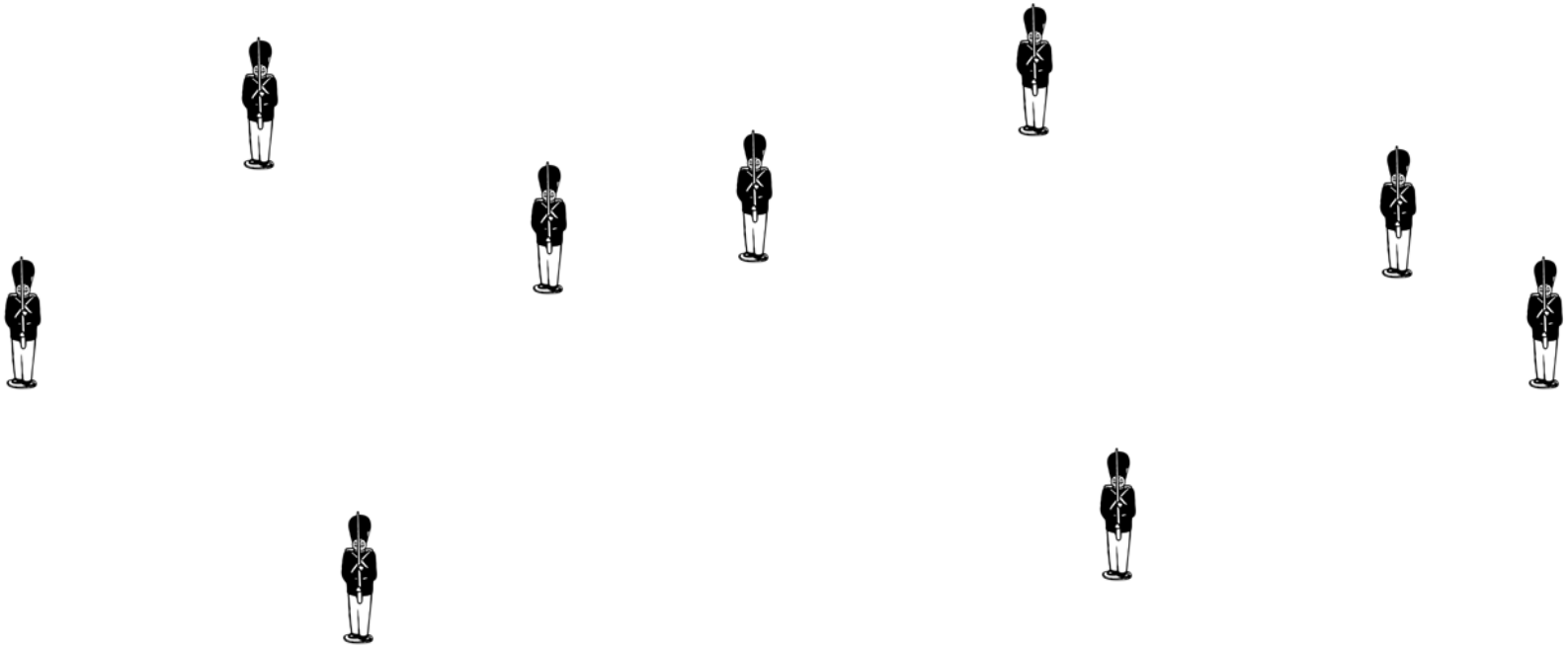
- **Framework**: common concept, different methods.
- **Efficient**: usually  $O(N)$  or  $O(N \log N)$  operations  
*The importance of efficient methods becomes greater as computers grow stronger!*
- **Iterative**: most nontrivial problems in our field cannot be solved directly efficiently.
- **Solving**: 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. Suppose we number the soldiers  $0$  to  $N$ , and that 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  $0$ .



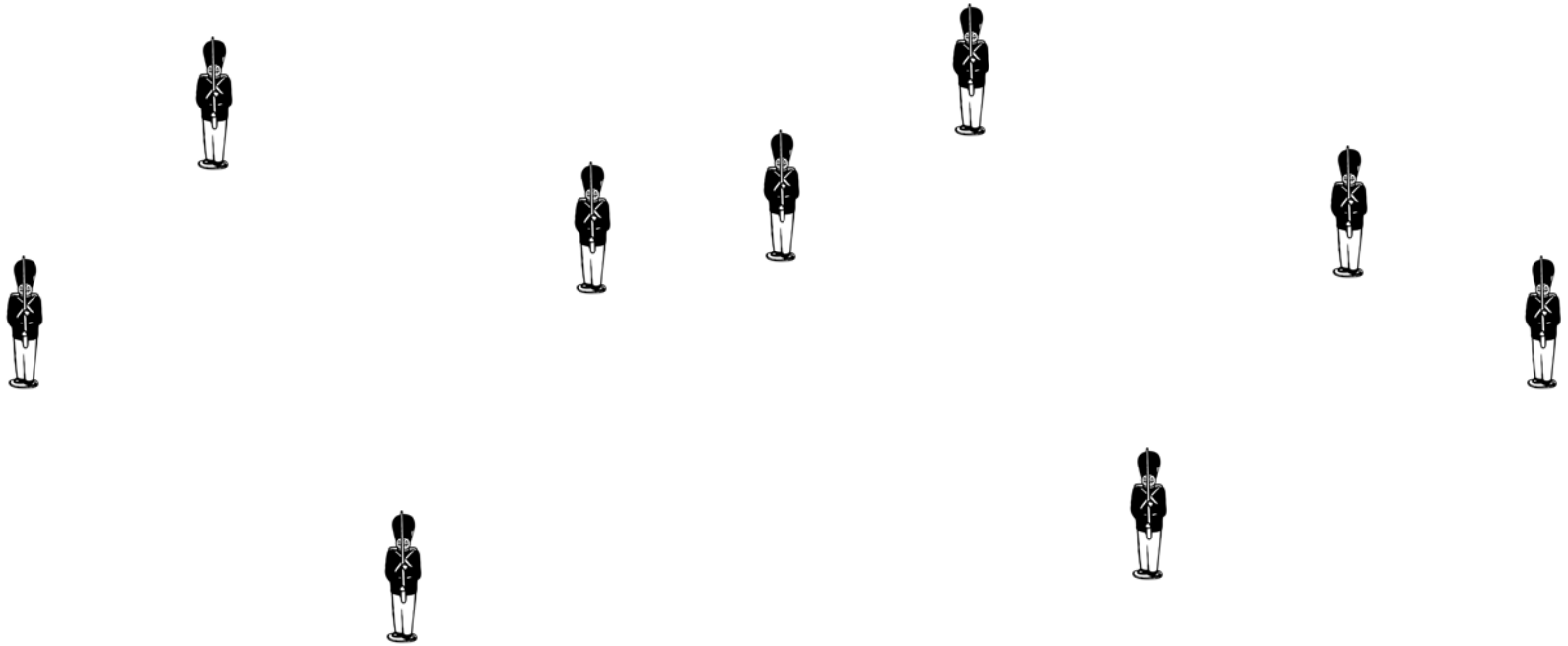


This method solves the problem **directly**, but it requires a high degree of **sophistication**: recognition of the extreme soldiers and some pretty fancy arithmetic.

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

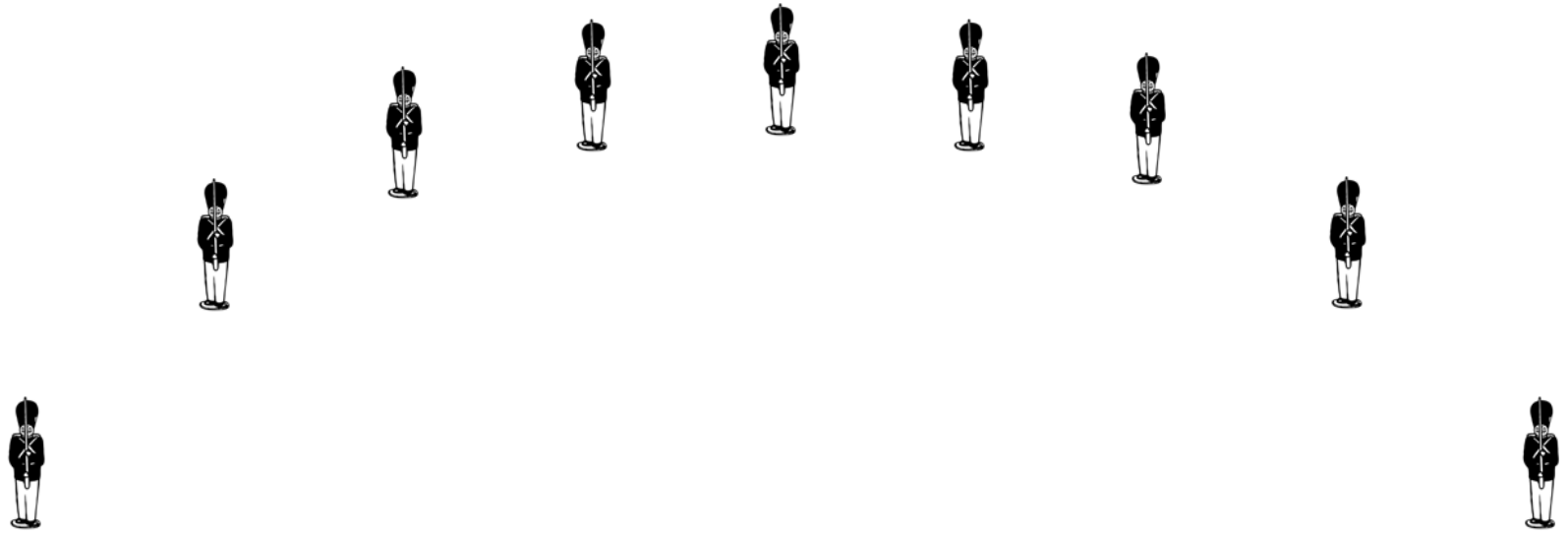
$$x_j^{(i)} = \frac{1}{2} \left( x_{j-1}^{(i-1)} + x_{j+1}^{(i-1)} \right)$$

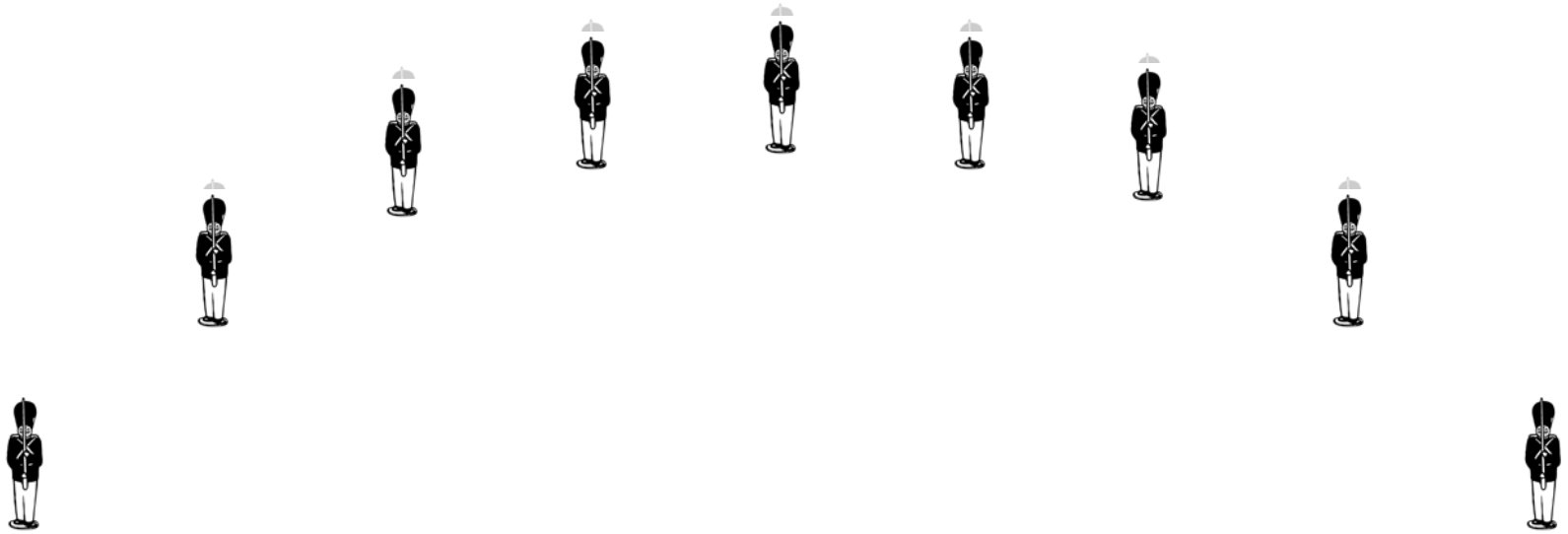
This is an iterative process. Each iteration brings us closer to the solution(?). The arithmetic is trivial.



A step in the right direction



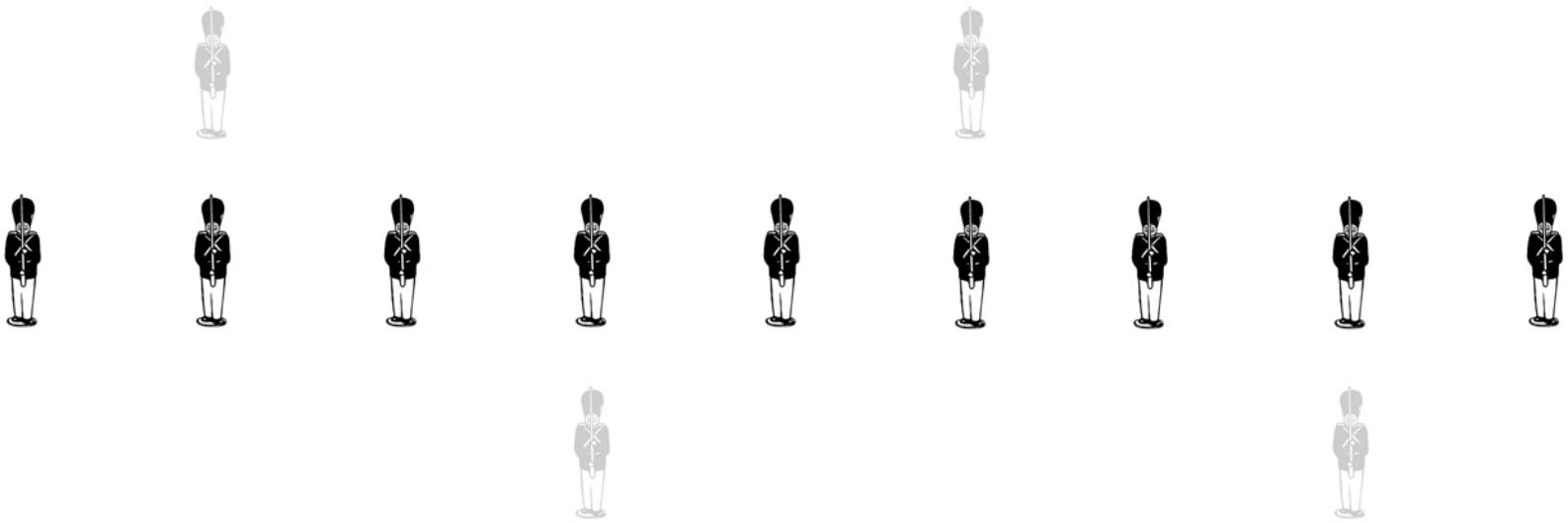




Slow convergence





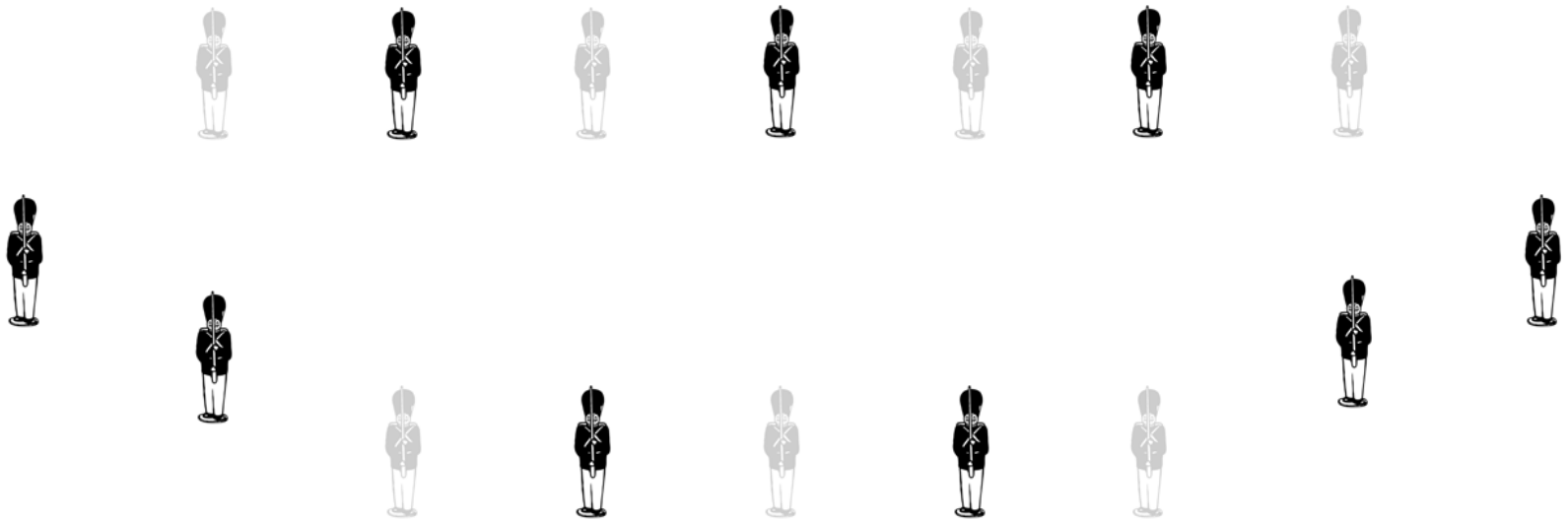


Fast convergence



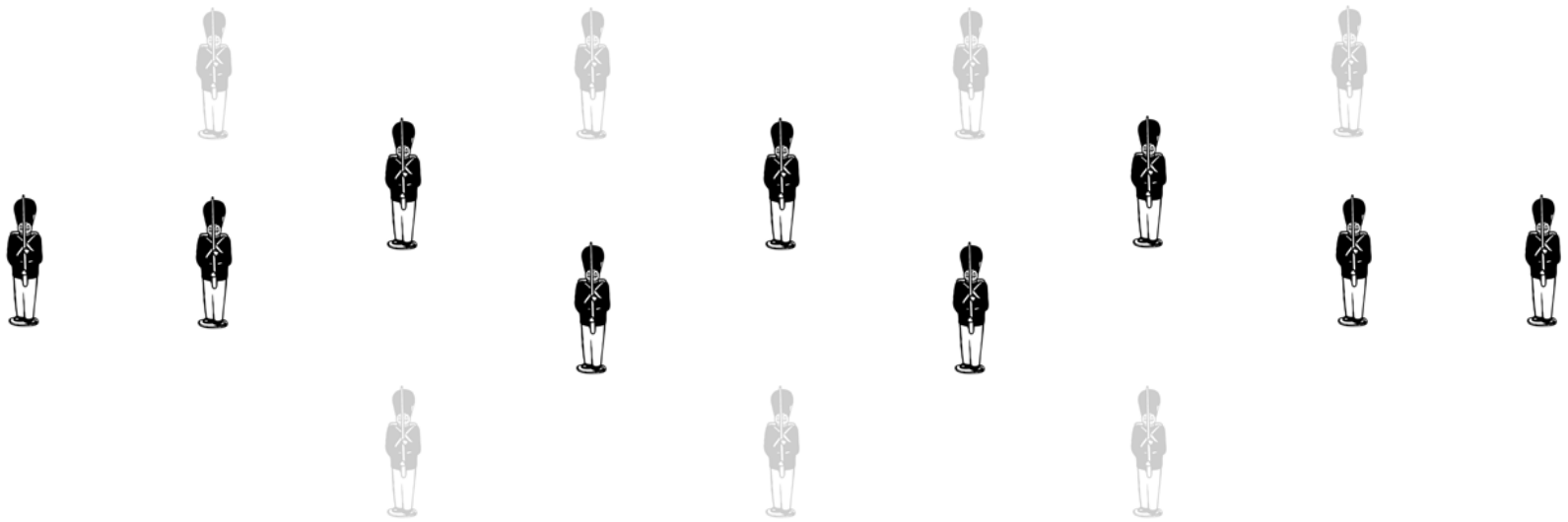






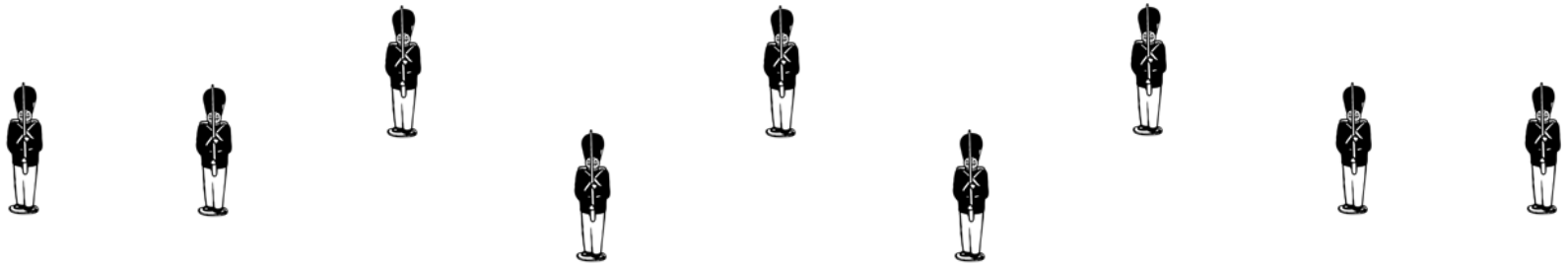
Slow convergence





Local solution: damping





Local solution: damping





Local solution: damping

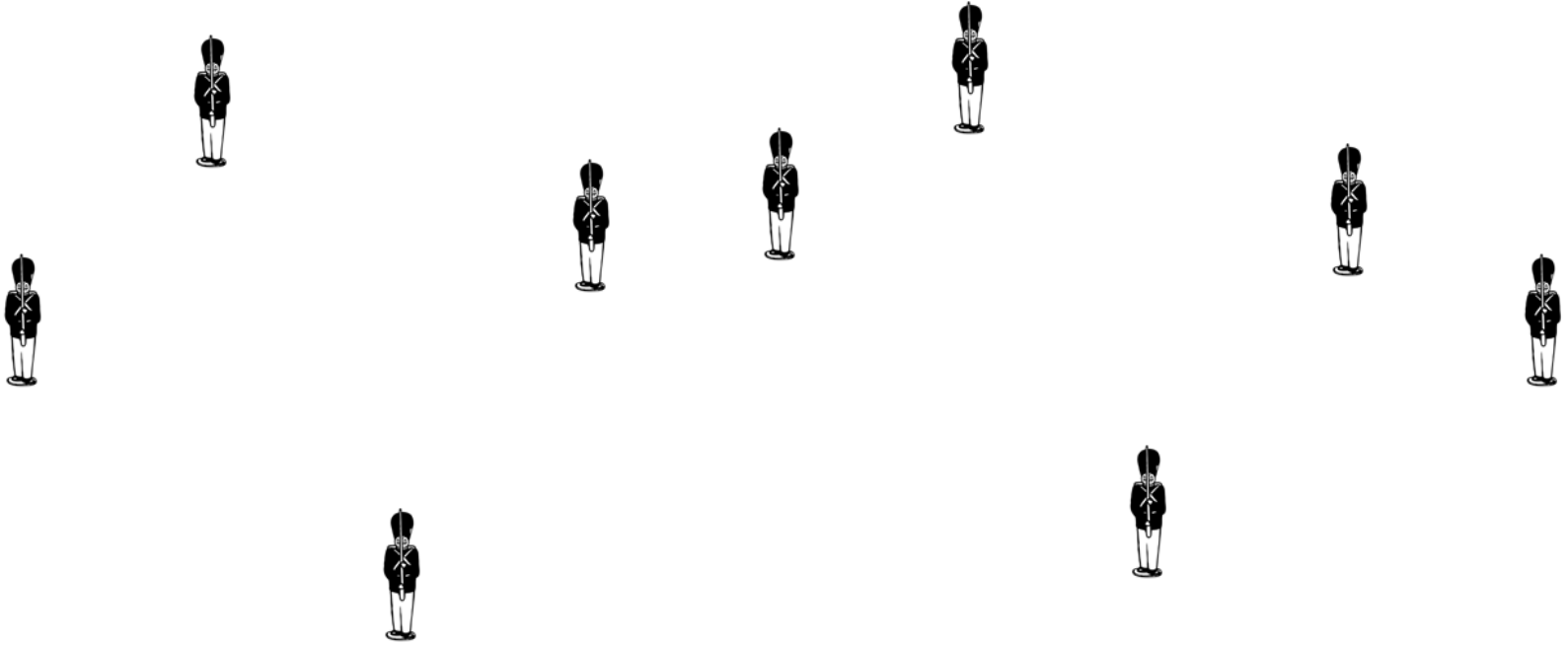




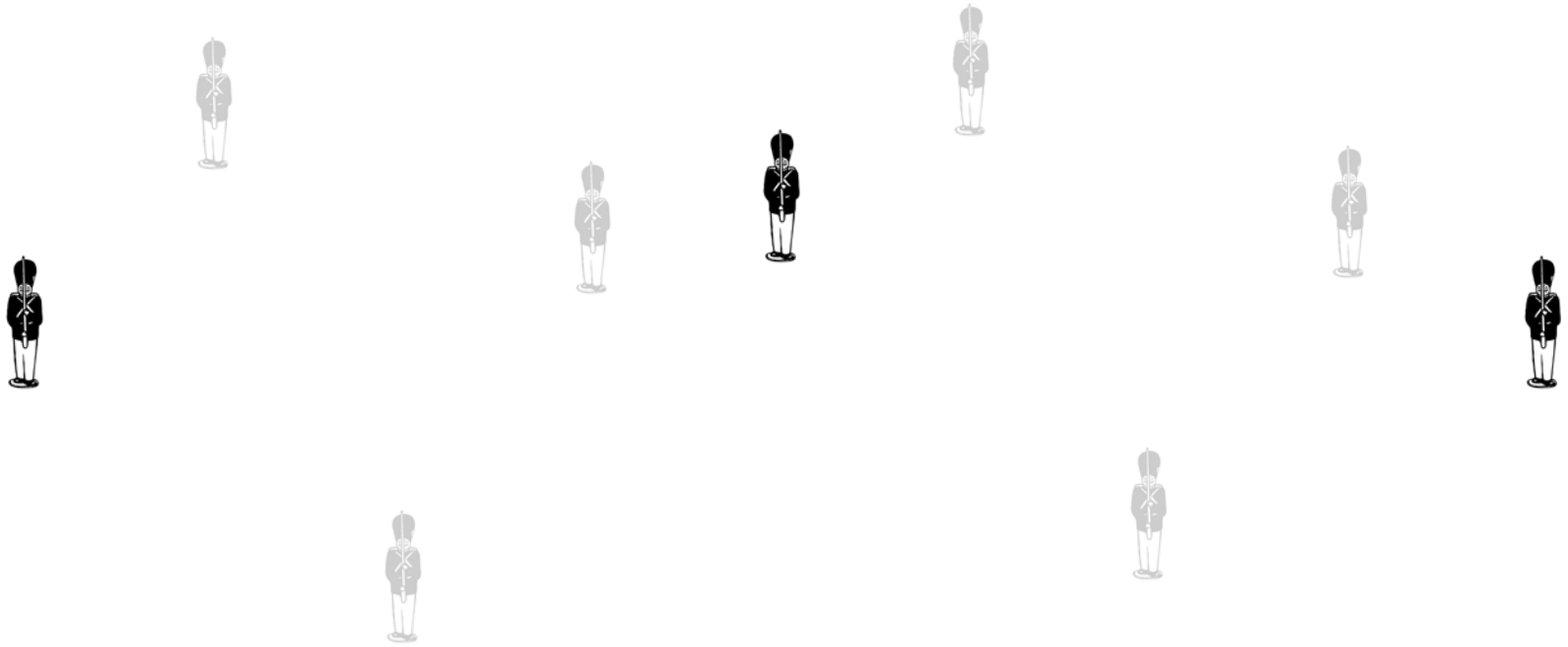
Local solution: damping



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

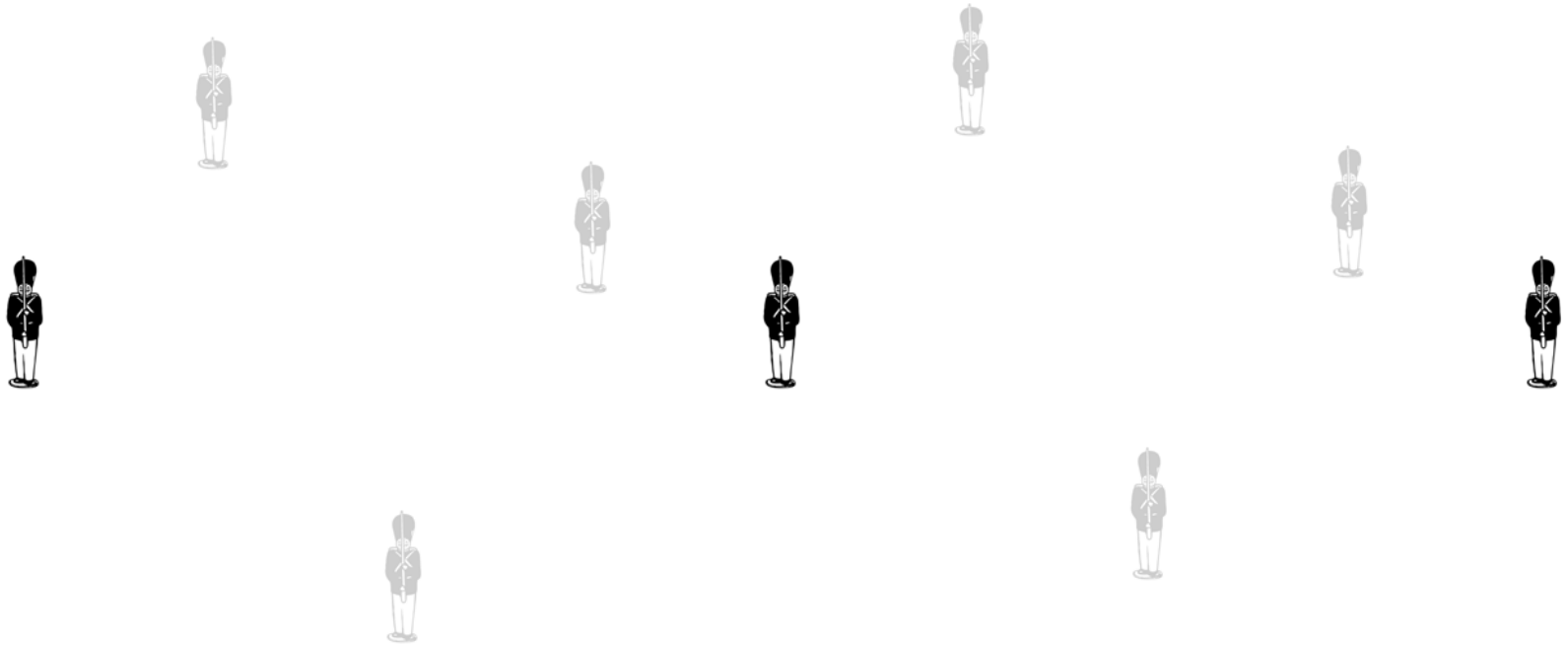






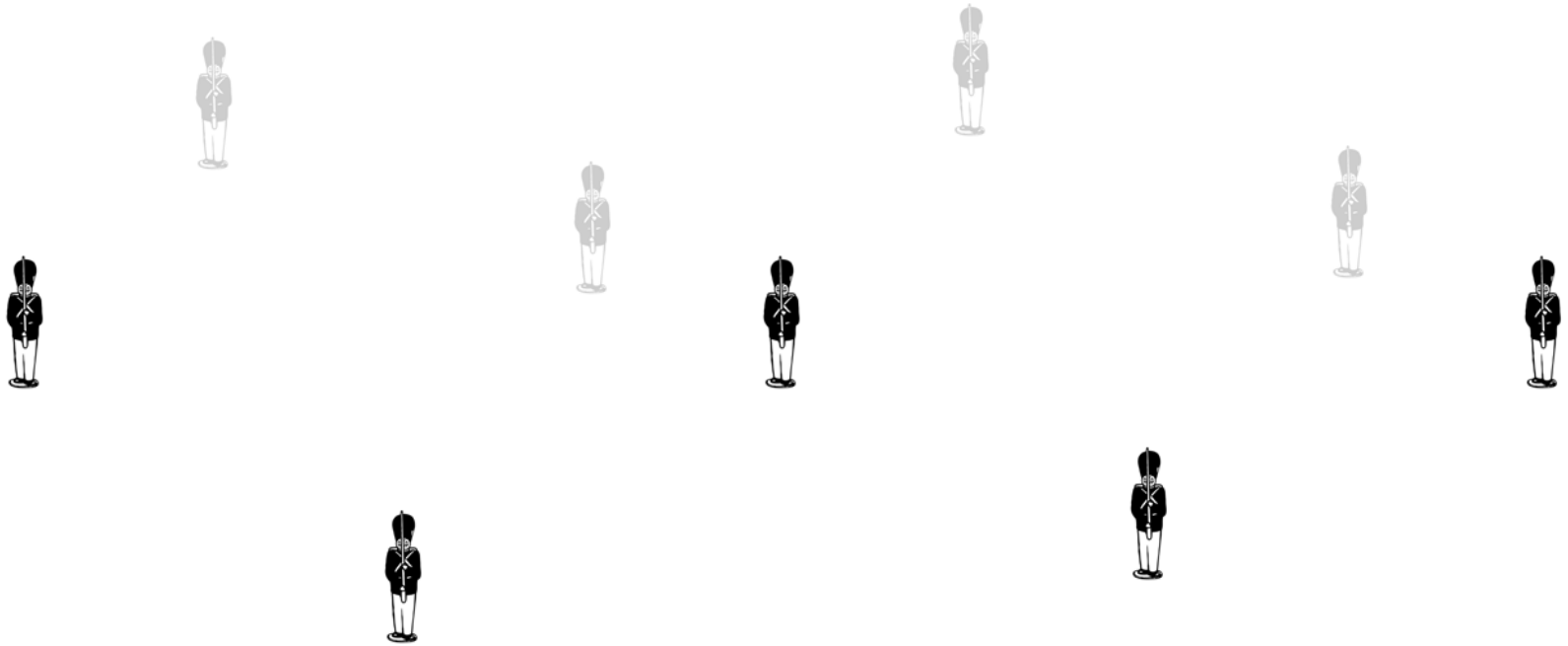
Large scale





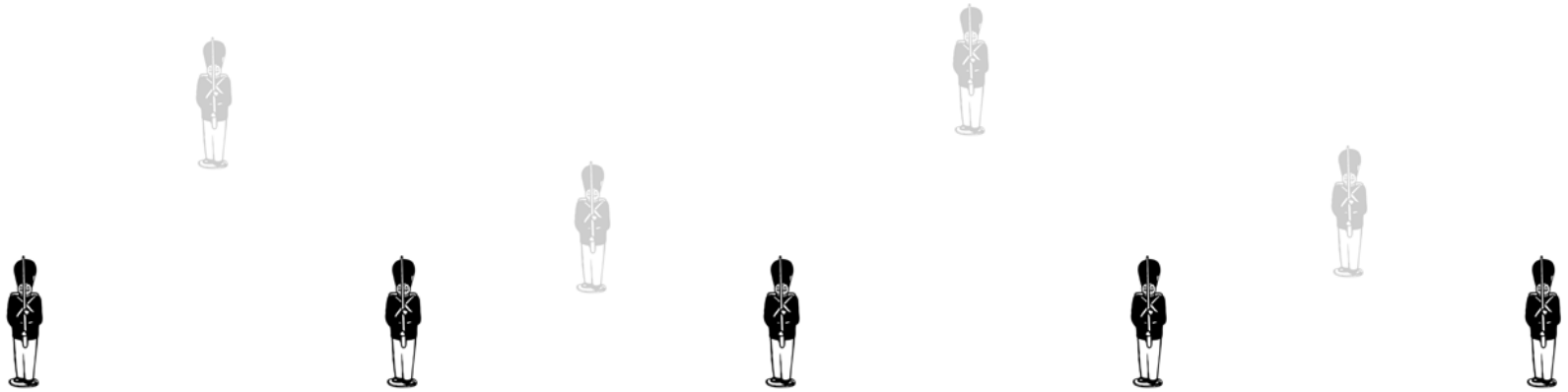
Large scale





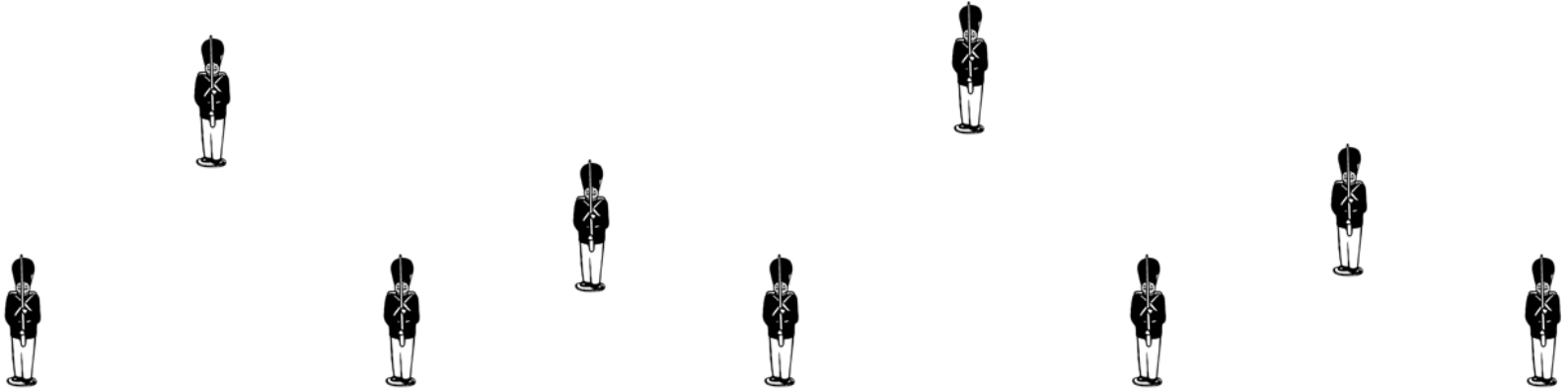
Intermediate scale





## Intermediate scale





Small scale







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

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

Since  $\mathbf{v}^k$  span the space  $\mathfrak{R}^{N-1}$ , 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:

$$\begin{aligned}\mathbf{x}^{(m)} &= \mathbf{S}\mathbf{x}^{(m-1)} = \mathbf{S}^2\mathbf{x}^{(m-2)} = \\ \dots &= \mathbf{S}^m\mathbf{x}^{(0)} = \mathbf{S}^m \sum_k c_k \mathbf{v}^k = \sum_k c_k \lambda_k^m \mathbf{v}^k\end{aligned}$$

Conclusion:

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

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

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

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

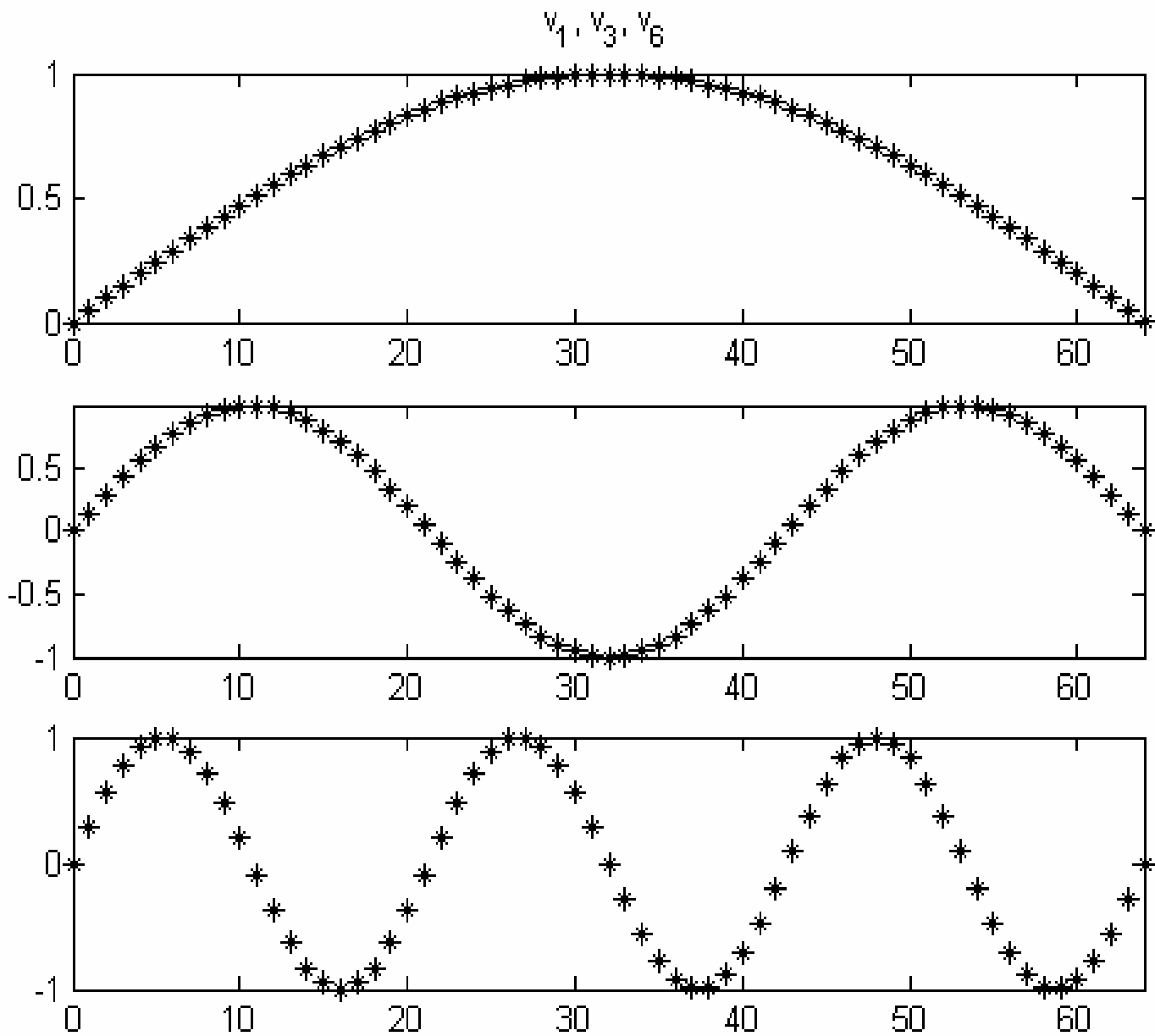
$$\lambda_k = \cos \left( \frac{k\pi}{N} \right),$$

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

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



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,

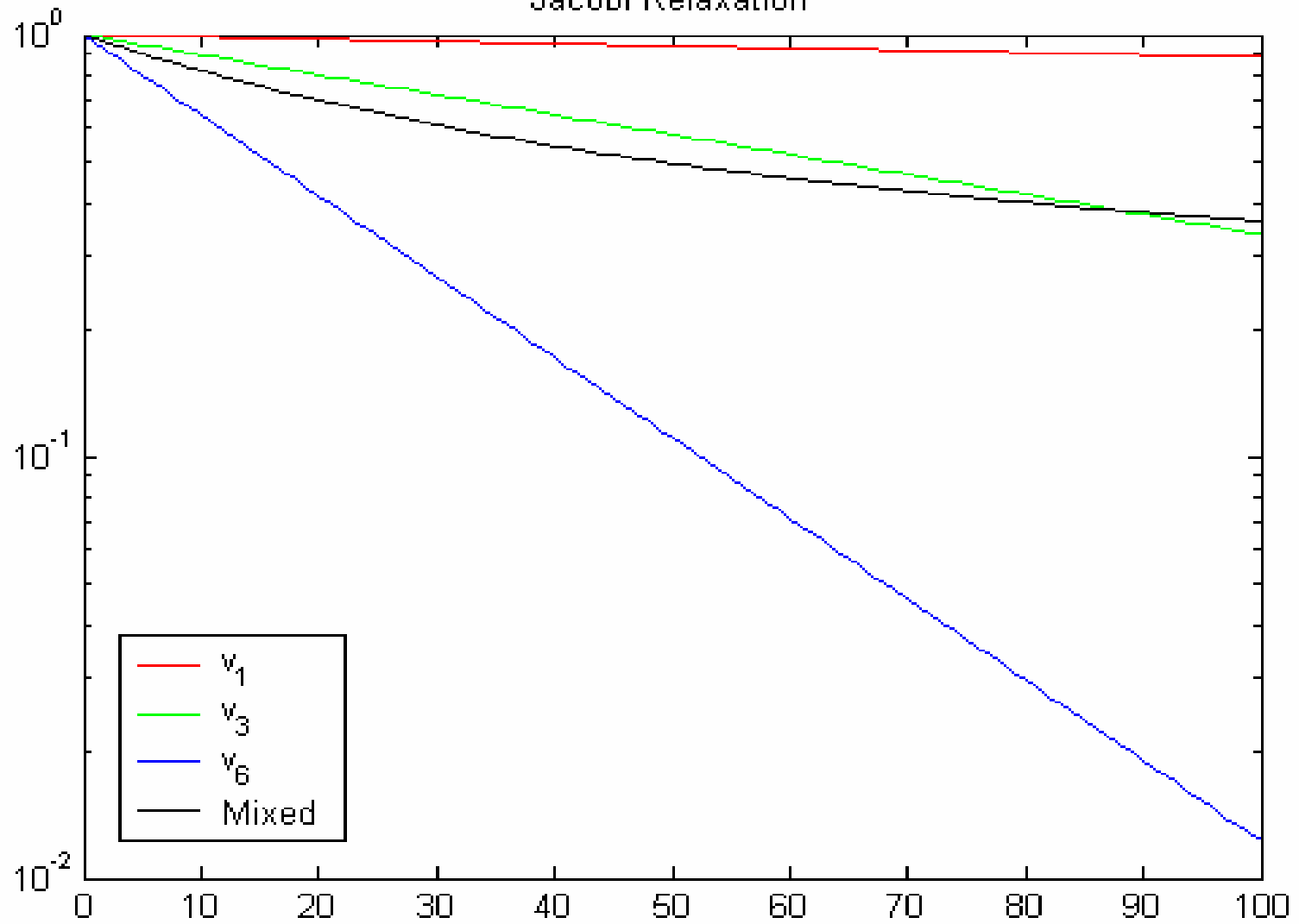
$$\lambda_k = \cos\left(\frac{k\pi}{N}\right) \approx 1 - \frac{1}{2}\left(\frac{k\pi}{N}\right)^2.$$

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



## 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  $\text{Log}_2(N)$  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?

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

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

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

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

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

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

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  $\omega$  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**,  $|\lambda_k^{(\omega)}| < 1$ .

## 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  $N$ :

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

i.e.,

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

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:

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

$$u(0) = u_0 ,$$

$$u(1) = u_1 .$$

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:

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

Derivation: by the **Taylor theorem**



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

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

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

$$u_0^h = u_0,$$

$$u_N^h = u_1.$$

In matrix form:

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

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

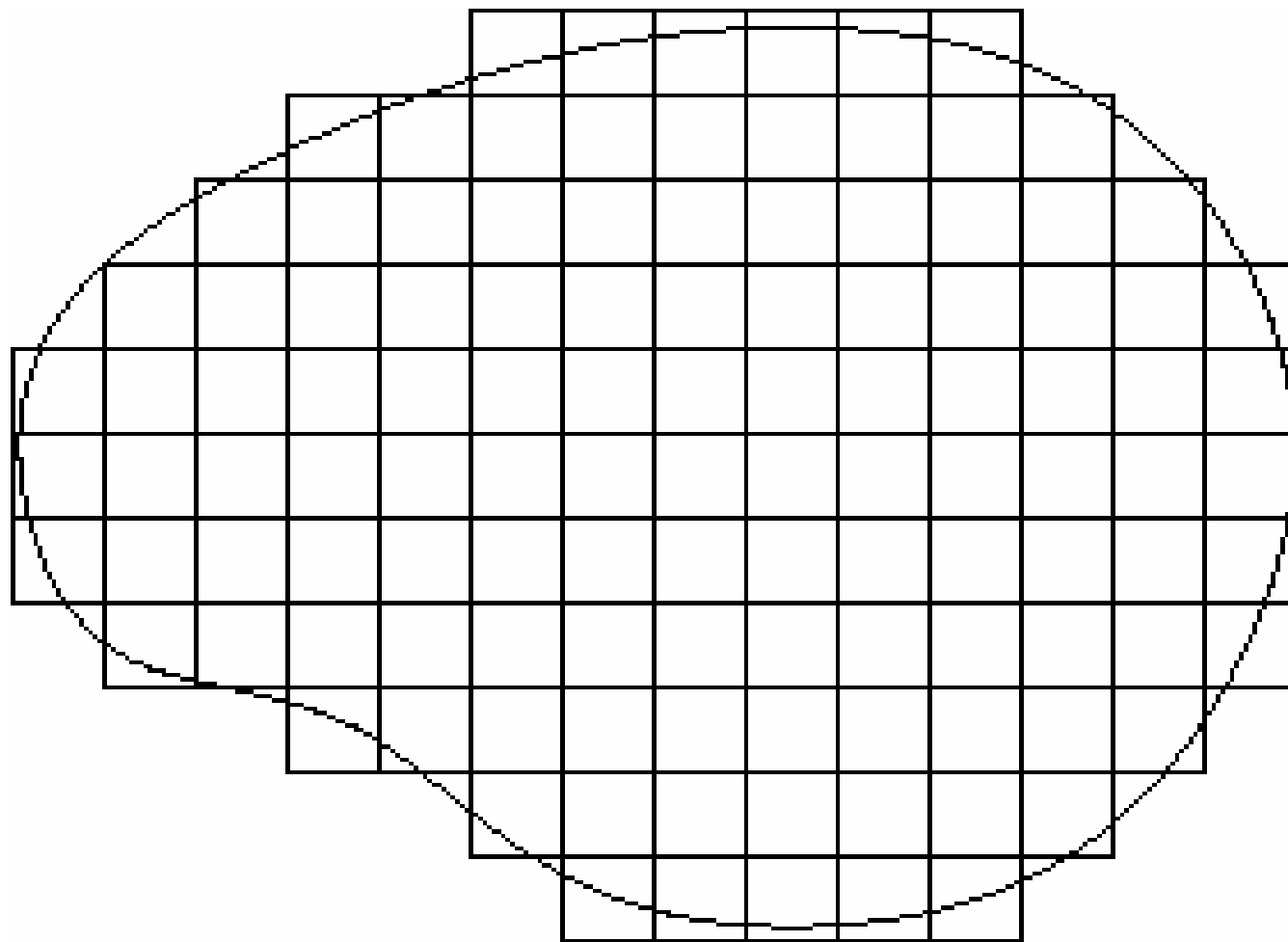
This is a **tridiagonal** system of equations which can be solved directly or iteratively.

## 2D Model Problem

Find  $u$  which satisfies:

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

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



$$\Omega^h$$

## 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:

$$L^h u_{i,j}^h = \frac{u_{i-1,j}^h - 2u_{i,j}^h + u_{i+1,j}^h}{h^2} + \frac{u_{i,j-1}^h - 2u_{i,j}^h + u_{i,j+1}^h}{h^2} = f_{i,j}^h \quad \text{in } \Omega^h \quad (5)$$

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

This yields a nonsingular linear system of equations for  $u_{i,j}^h$  (the discrete operator satisfies a **maximum principle**.)

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

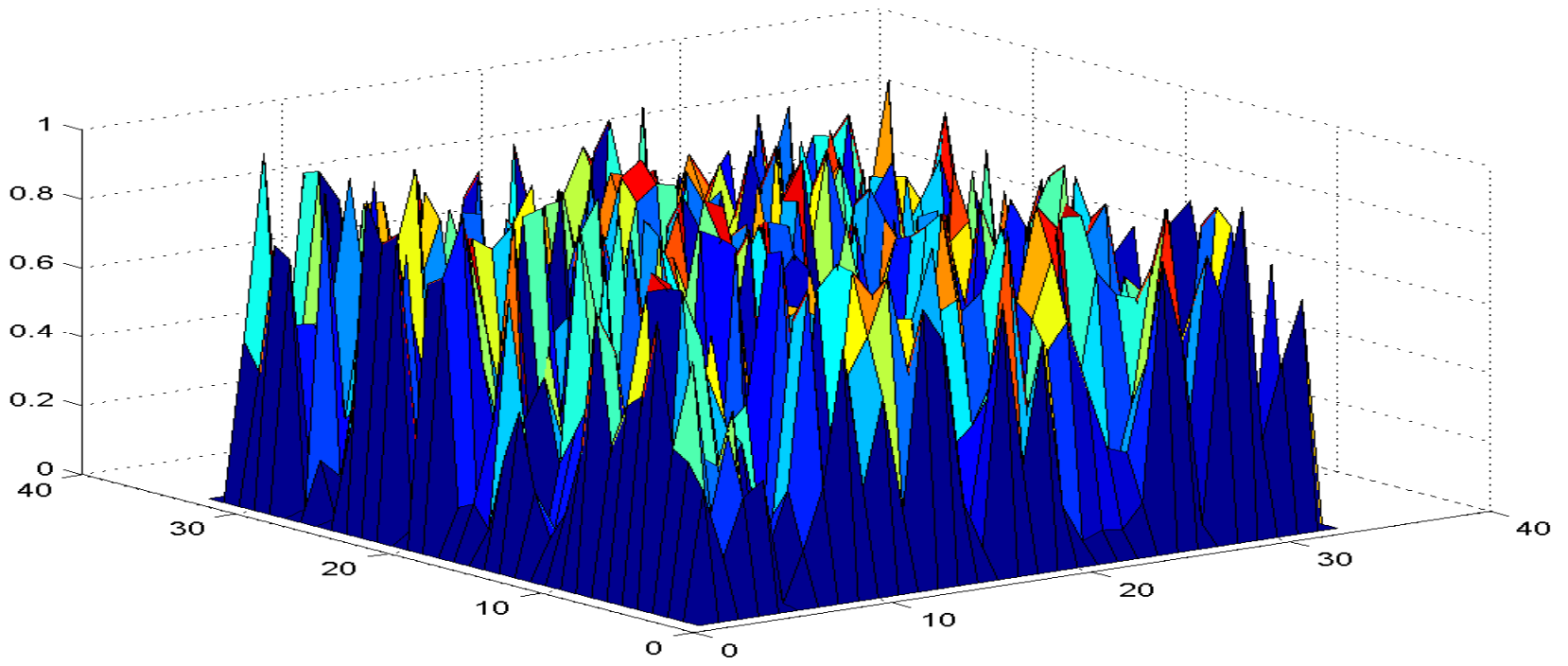
## Gauss-Seidel (GS) Relaxation:

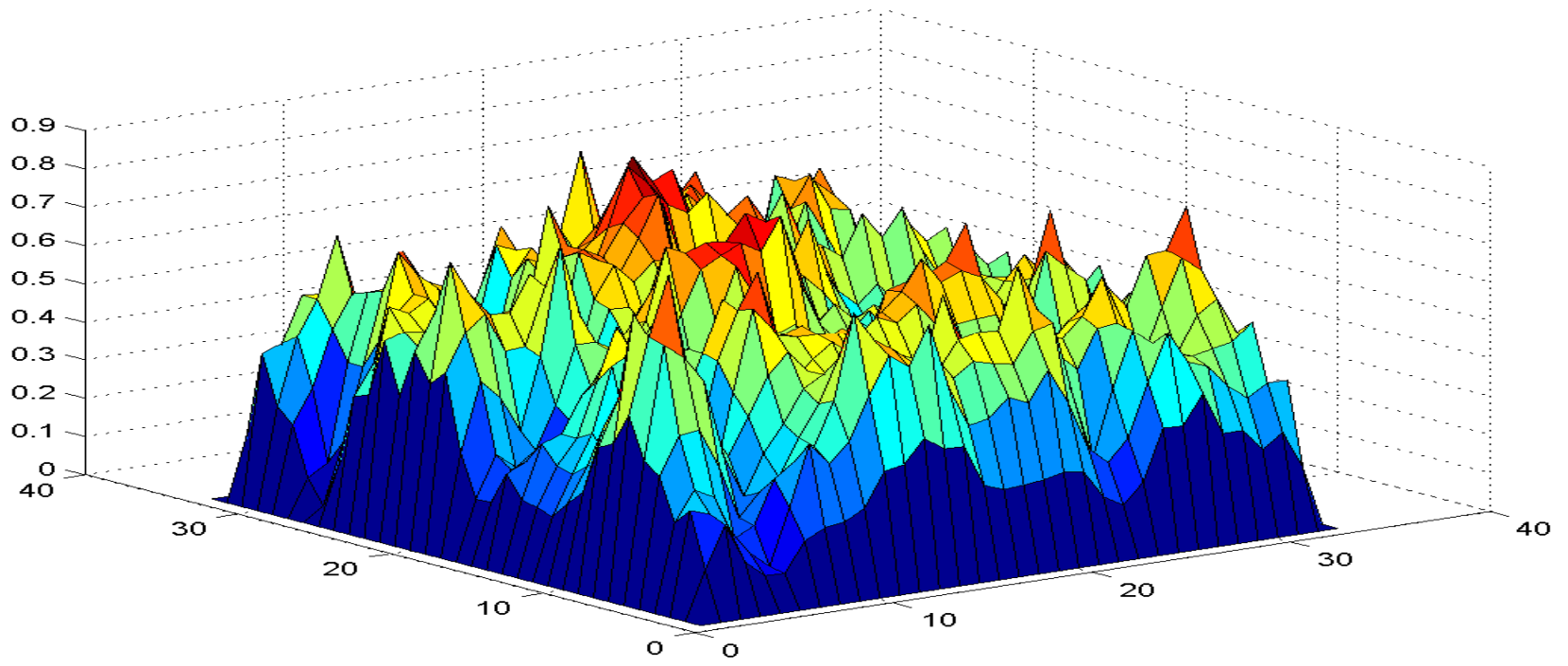
1. Choose initial guess,  $\tilde{u}^h$ .
2. Repeat until some convergence criterion is satisfied  
{  
Scan all variables in some prescribed order, and change each variable  $\tilde{u}_{i,j}^h$  in turn so as to satisfy the  $(i,j)$ th equation.  
}

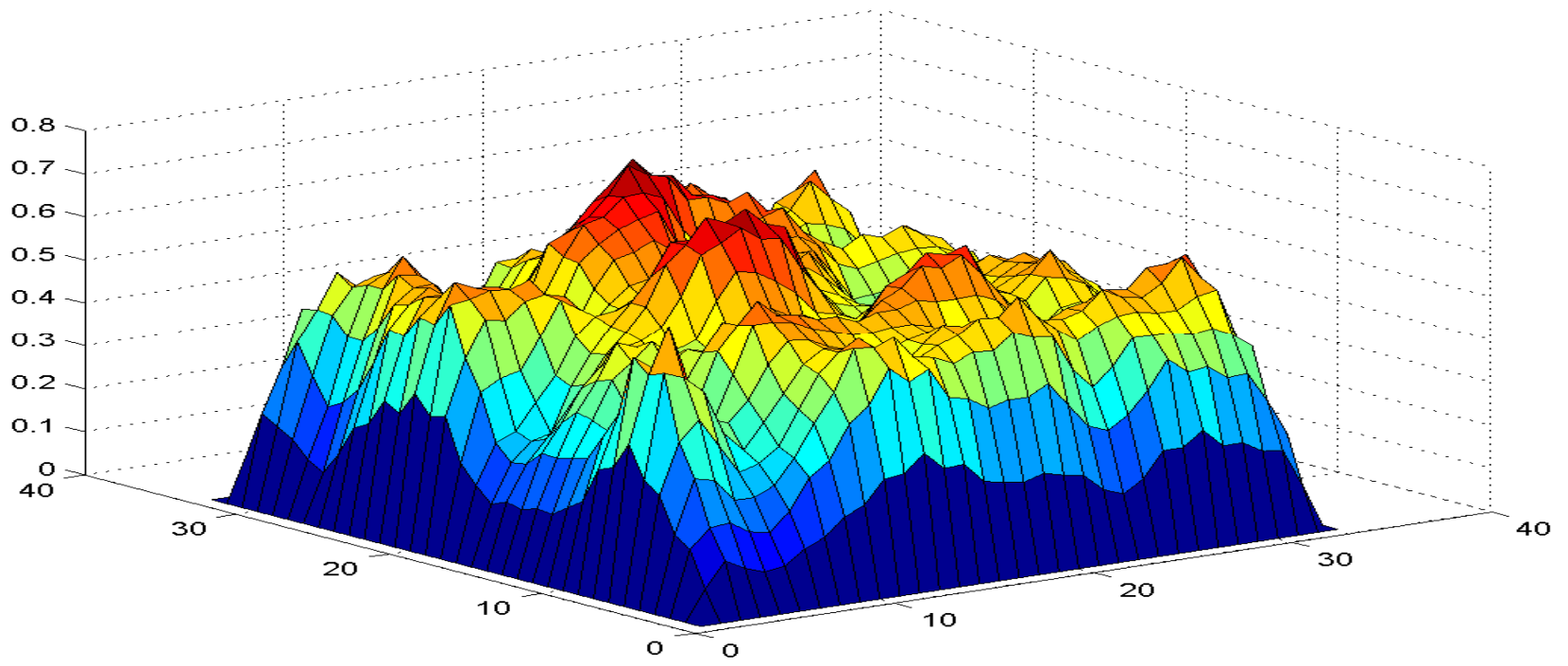
Observation: GS is a local process, because only near neighbors appear in each equation. Hence, it may be efficient for eliminating errors which can be detected locally. But large-scale ("smooth") errors are eliminated very slowly.

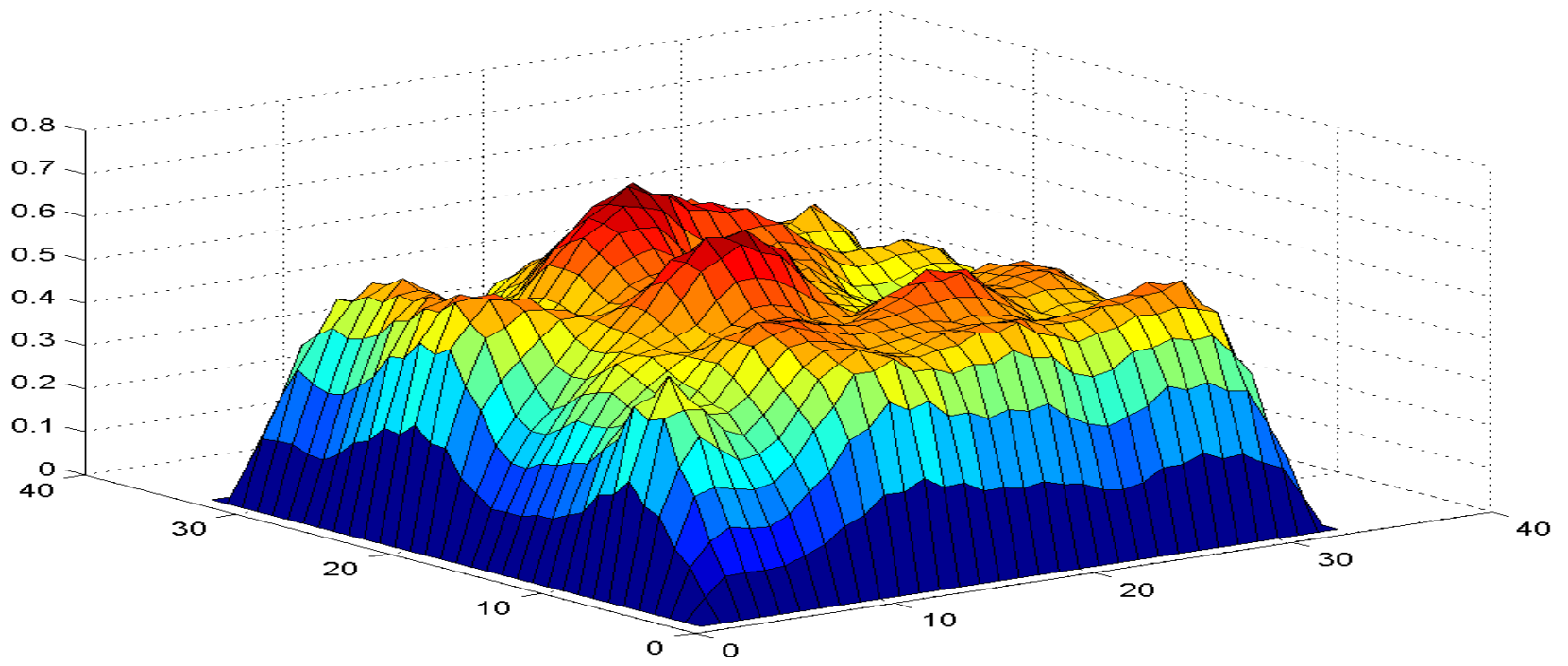
(The difference between GS and Jacobi is that old neighboring values are used in Jacobi, while the most updated values are used in GS.)

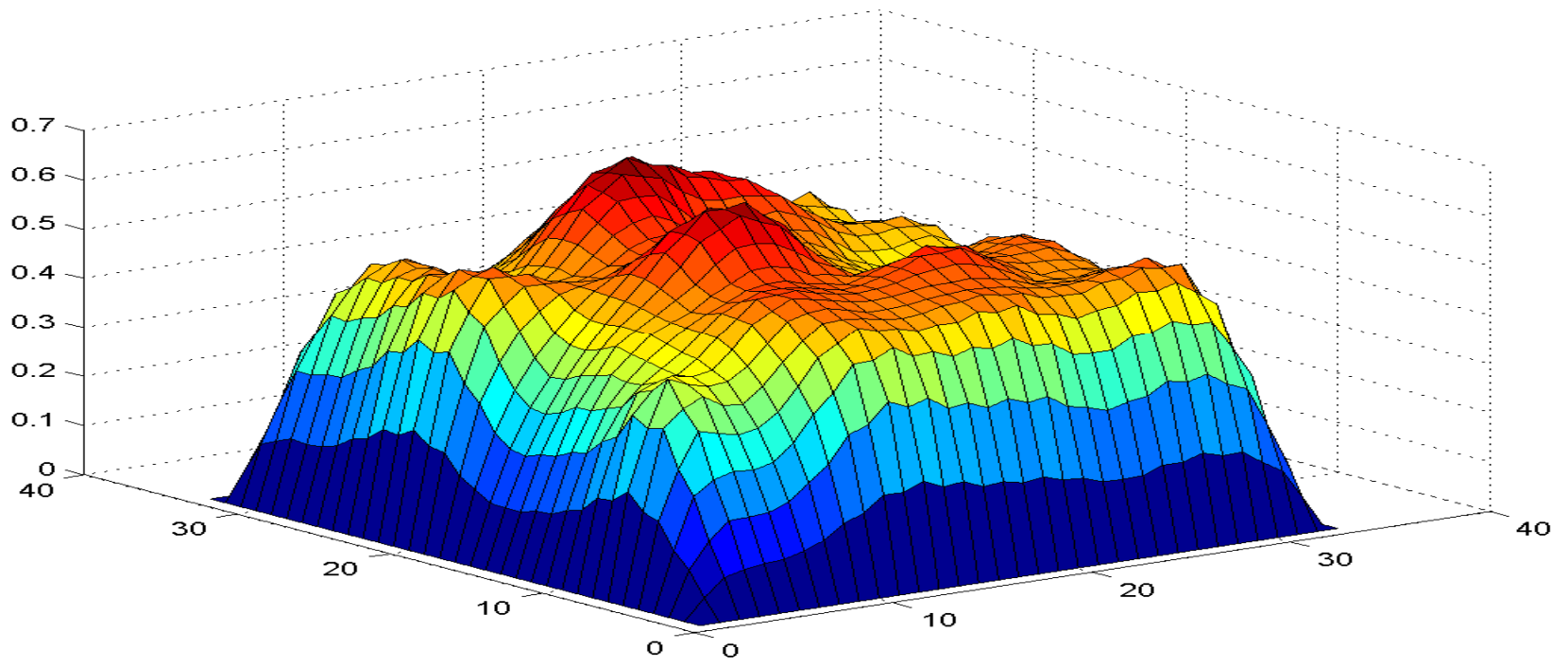


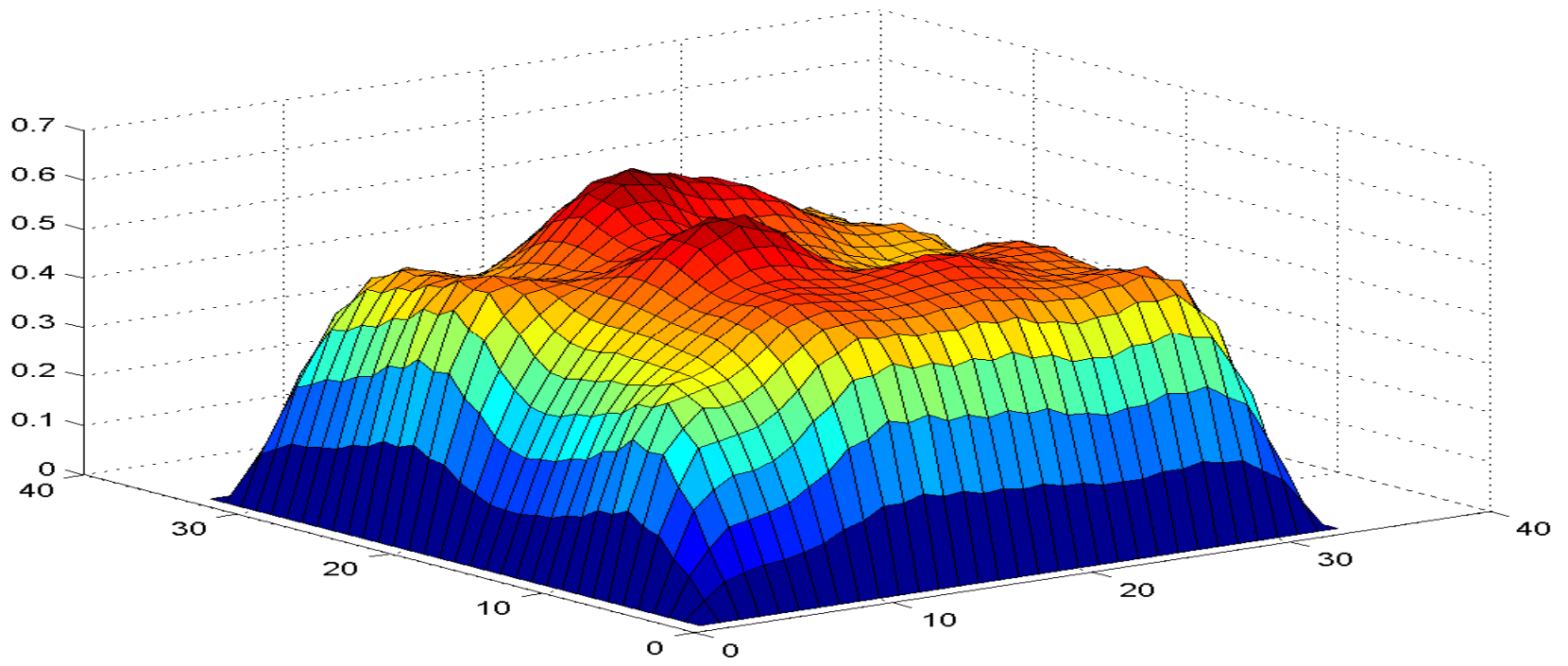


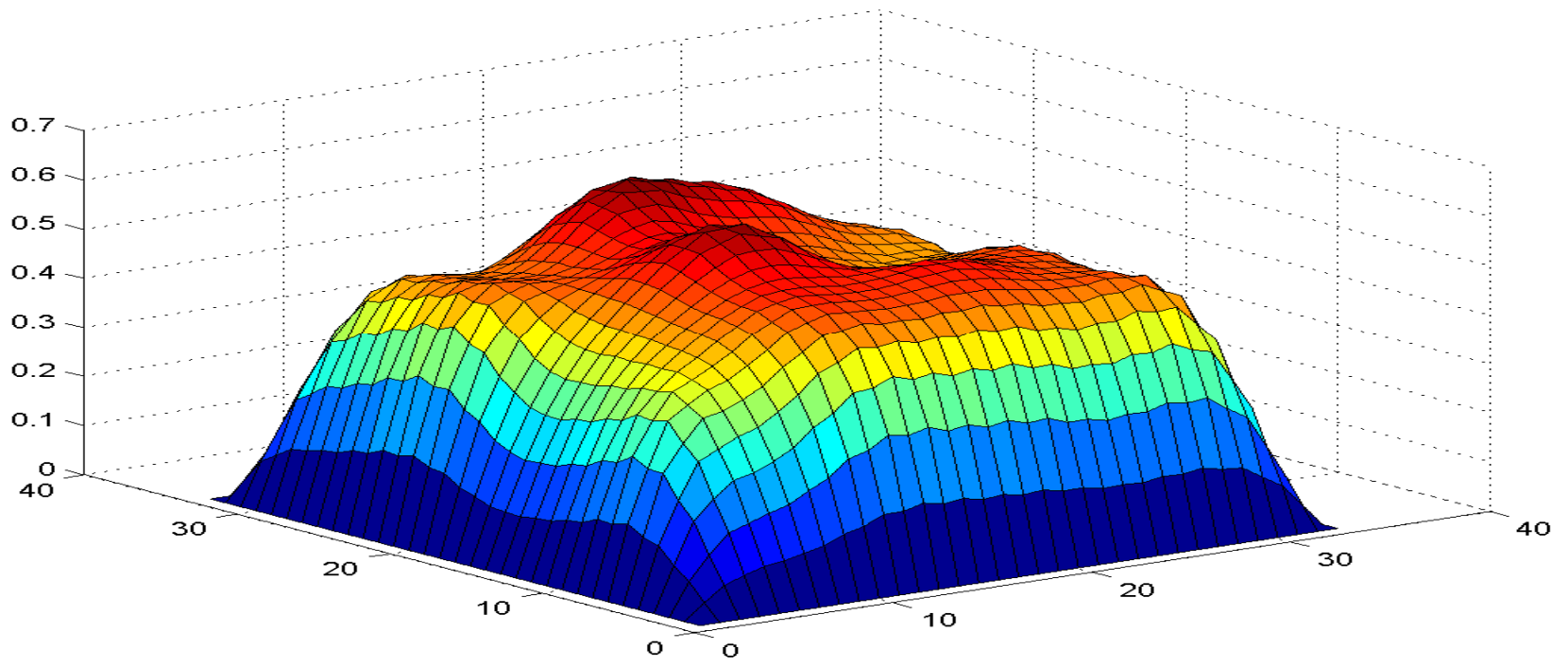


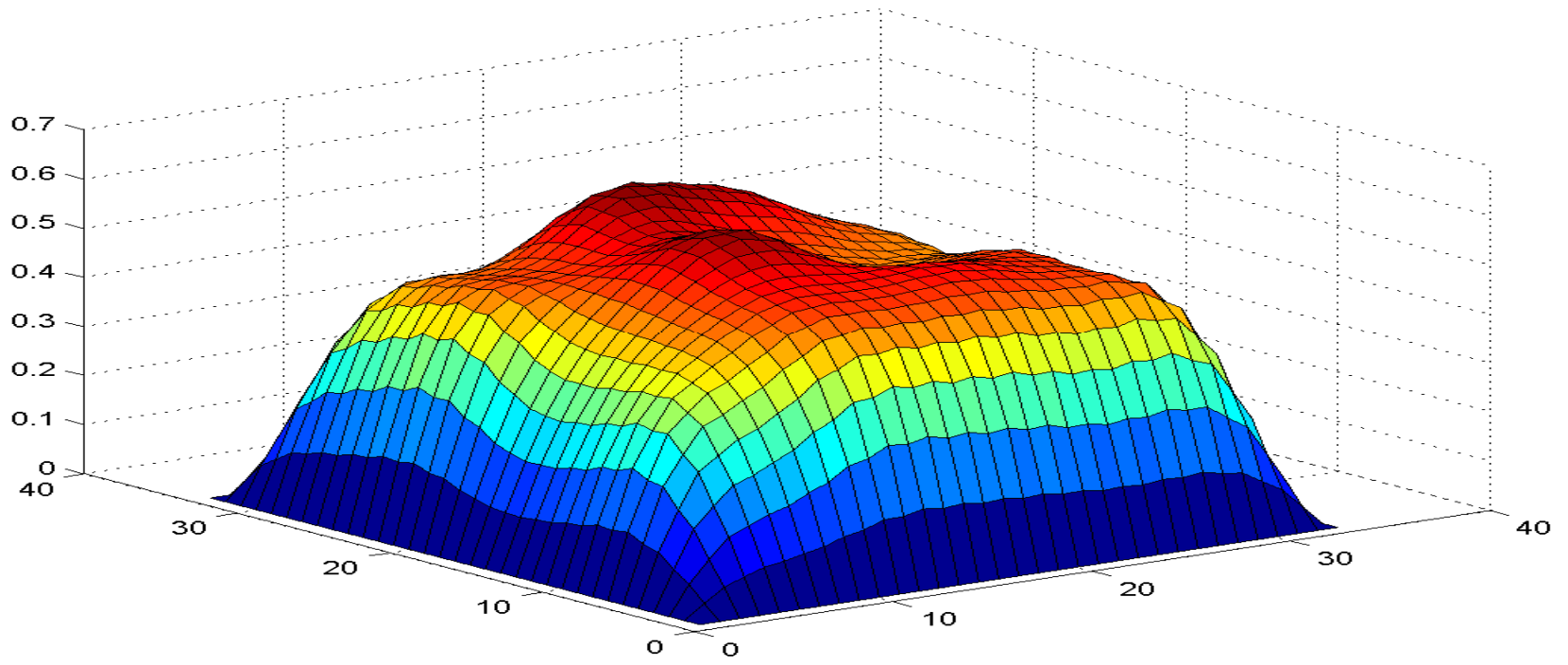




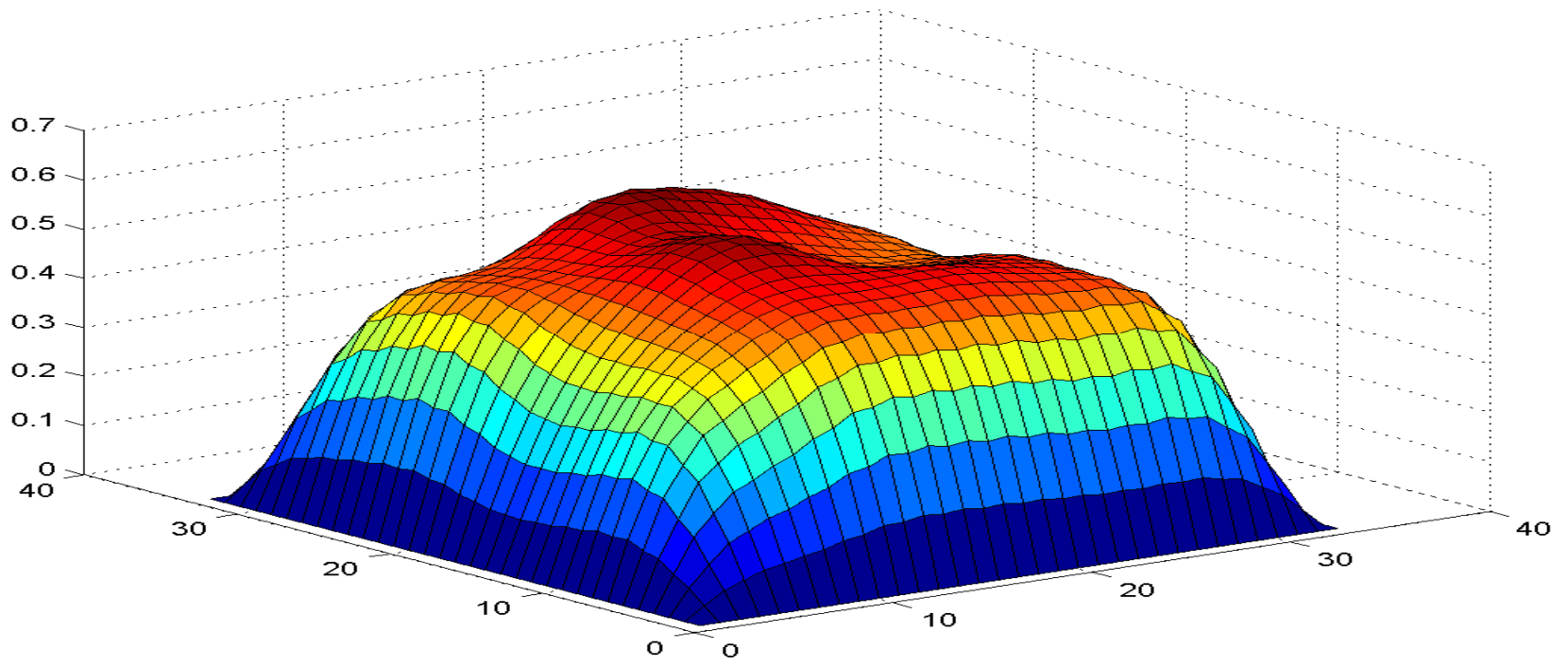


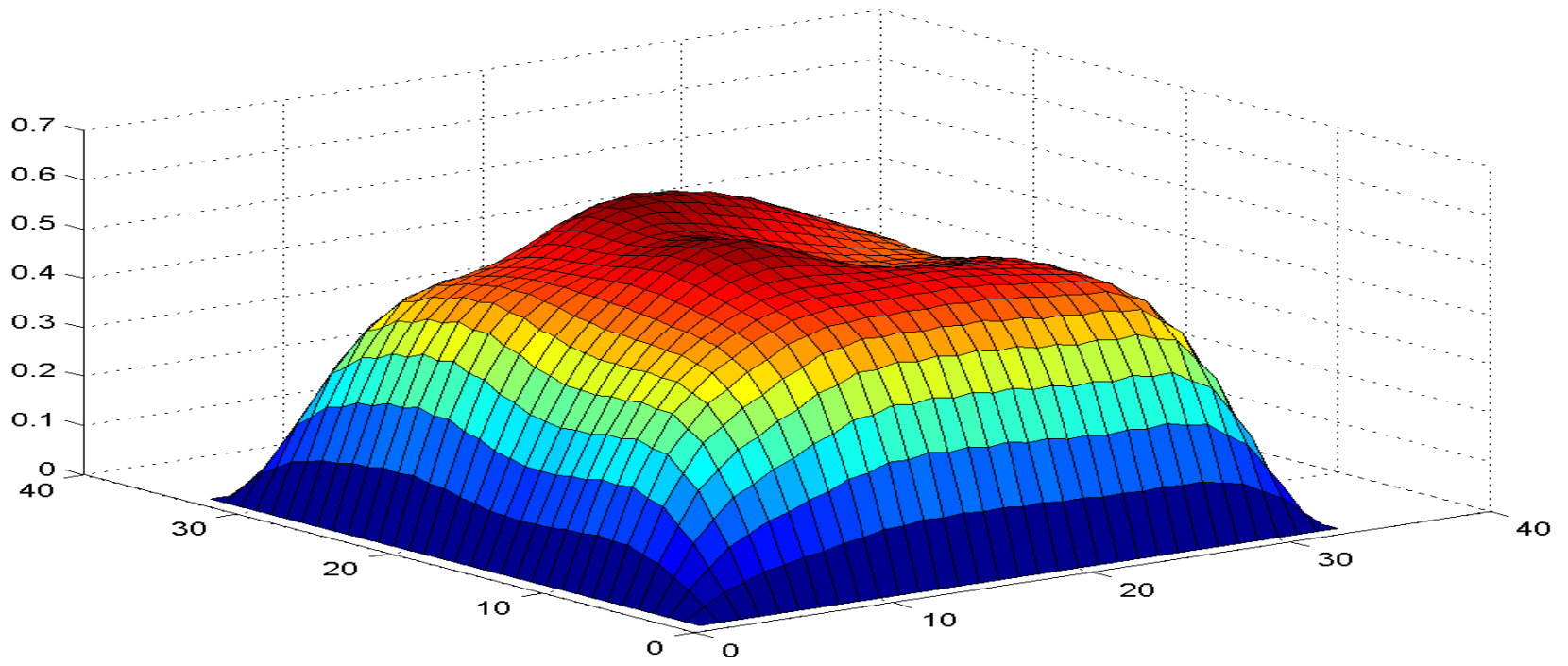


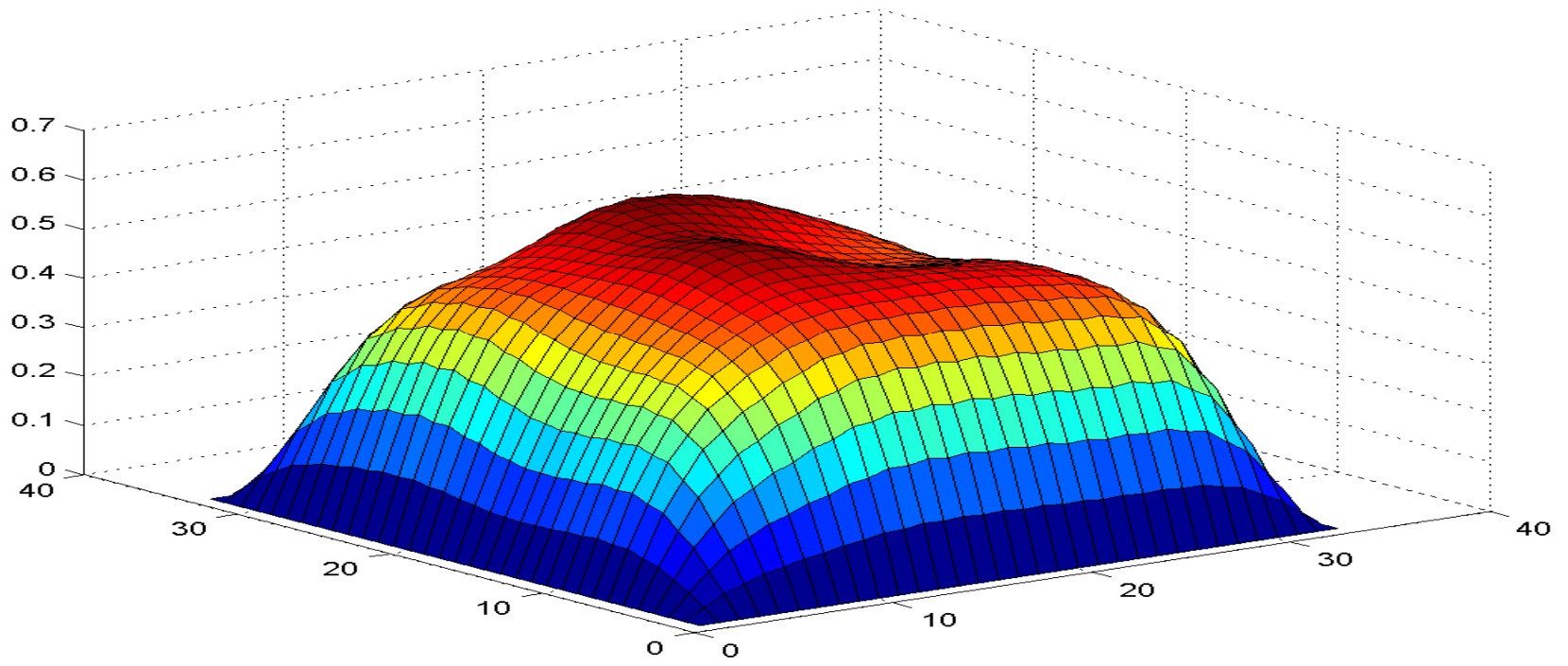


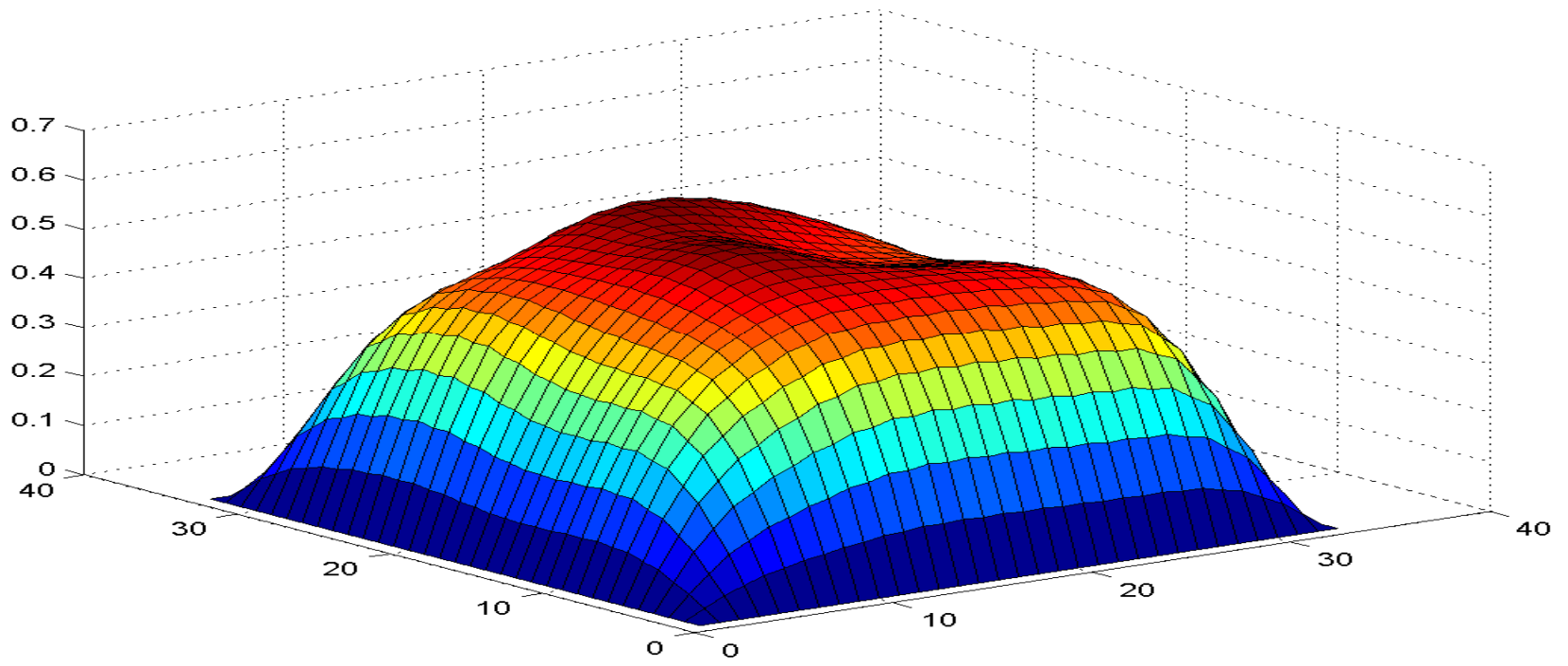


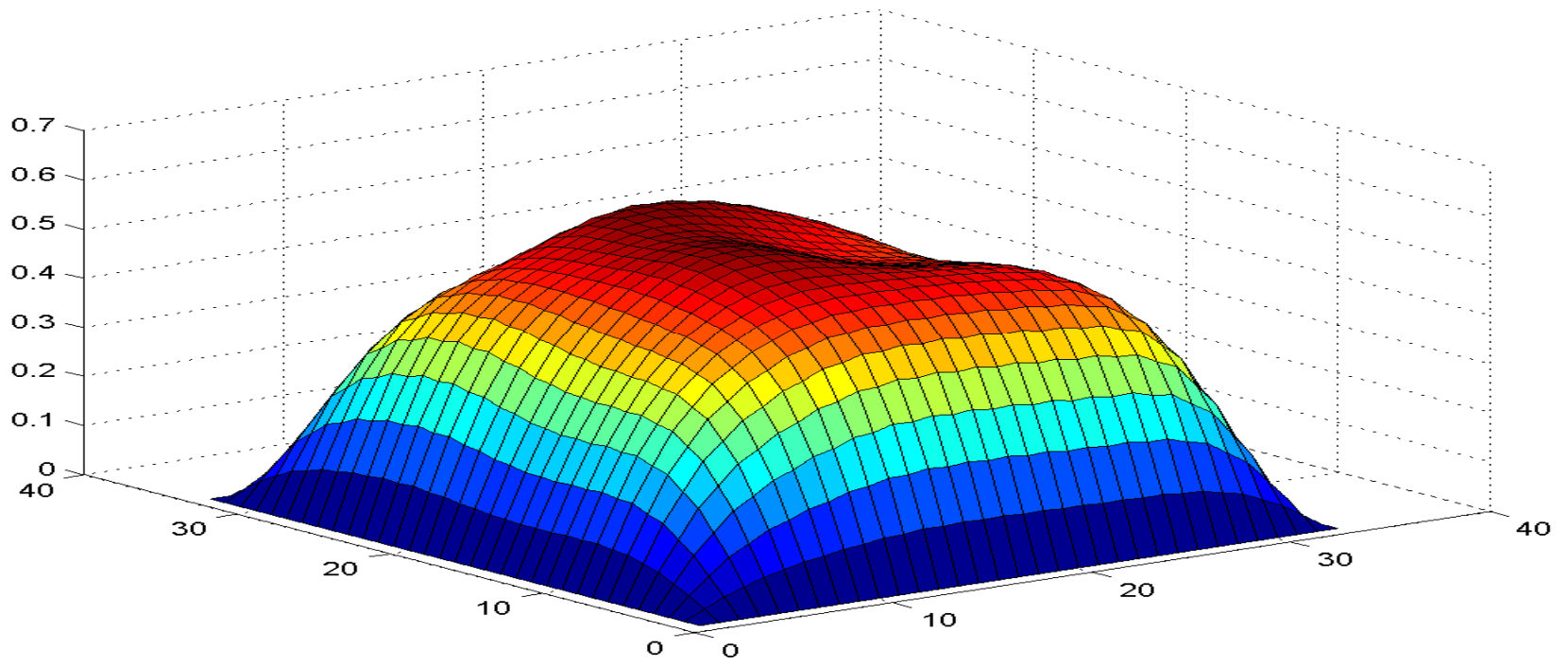


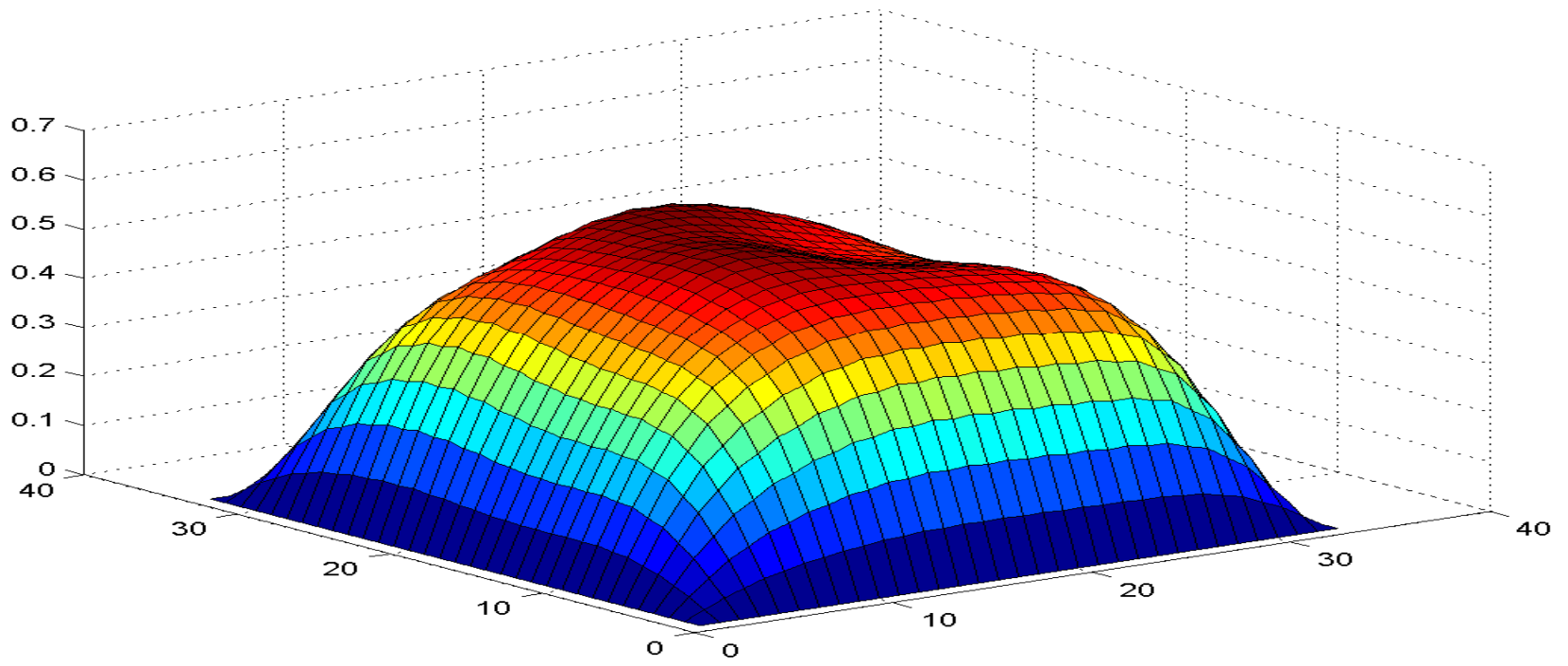


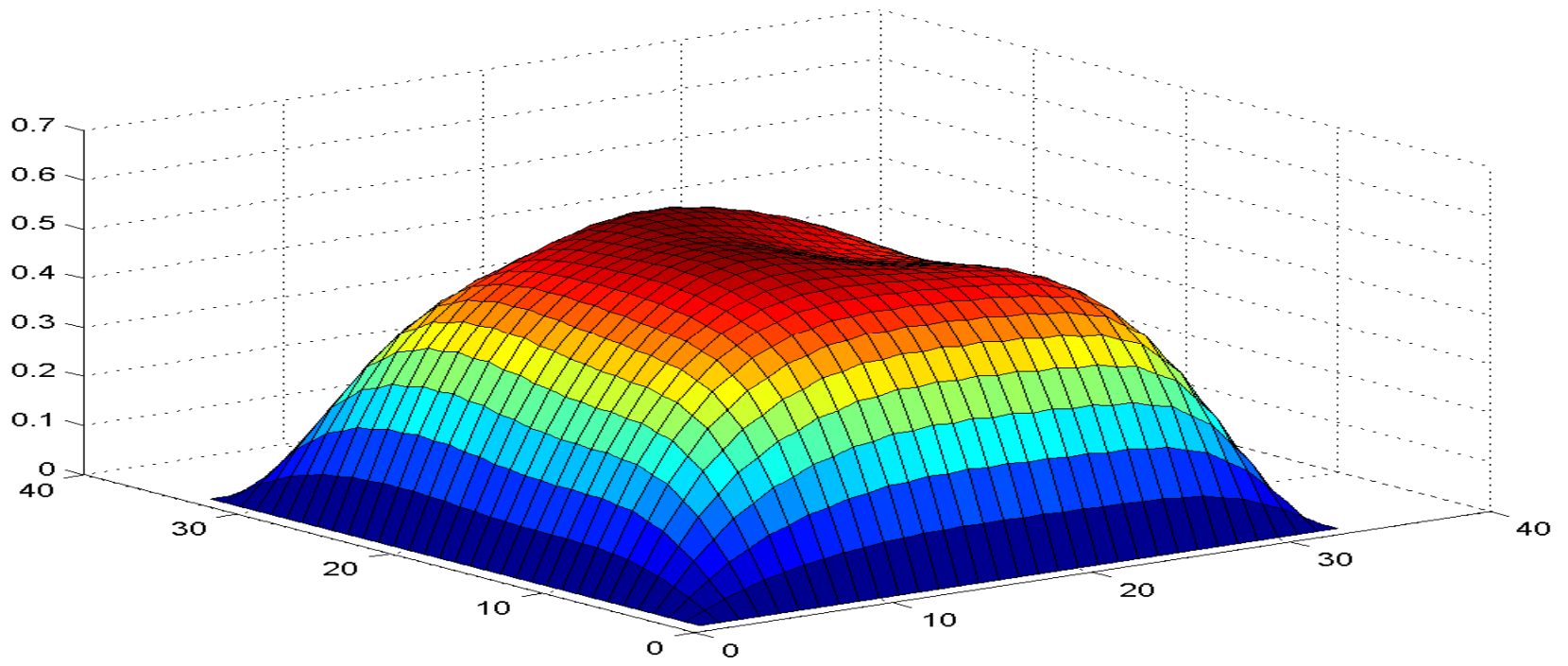


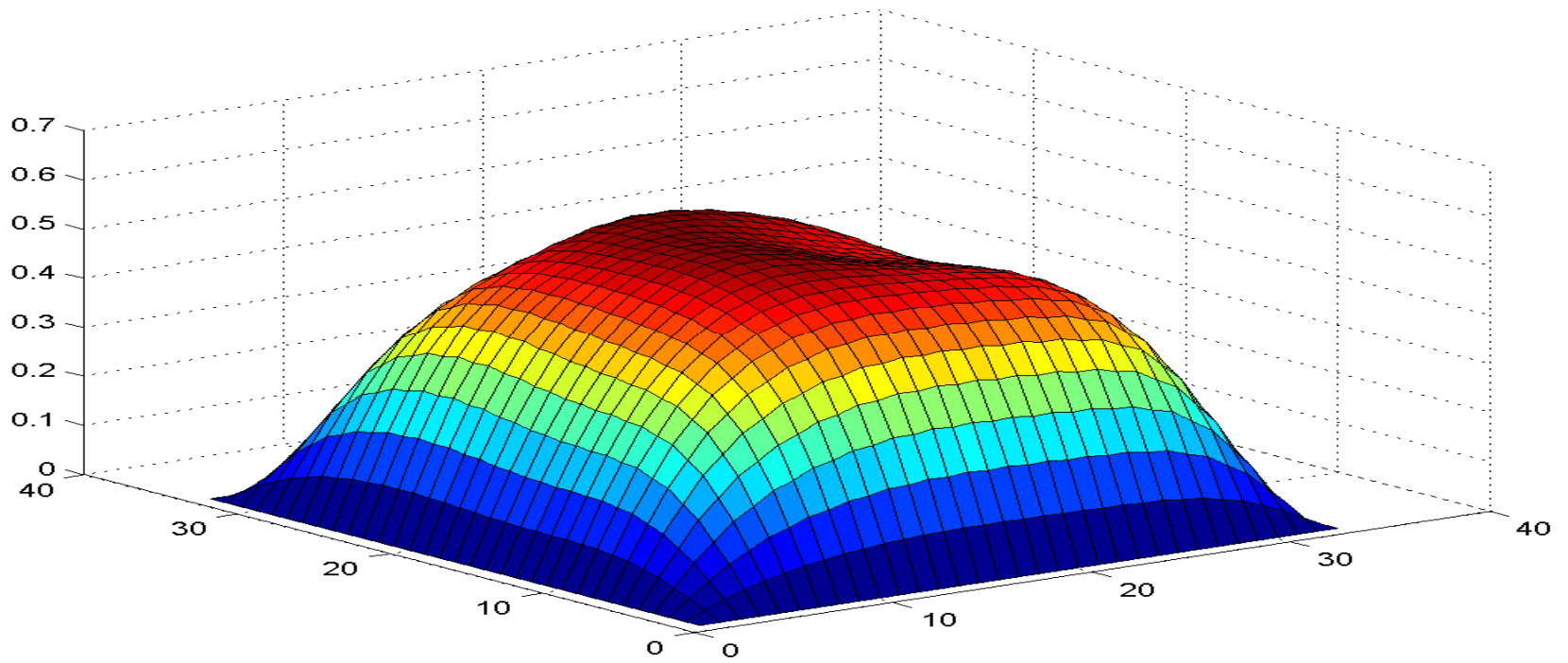




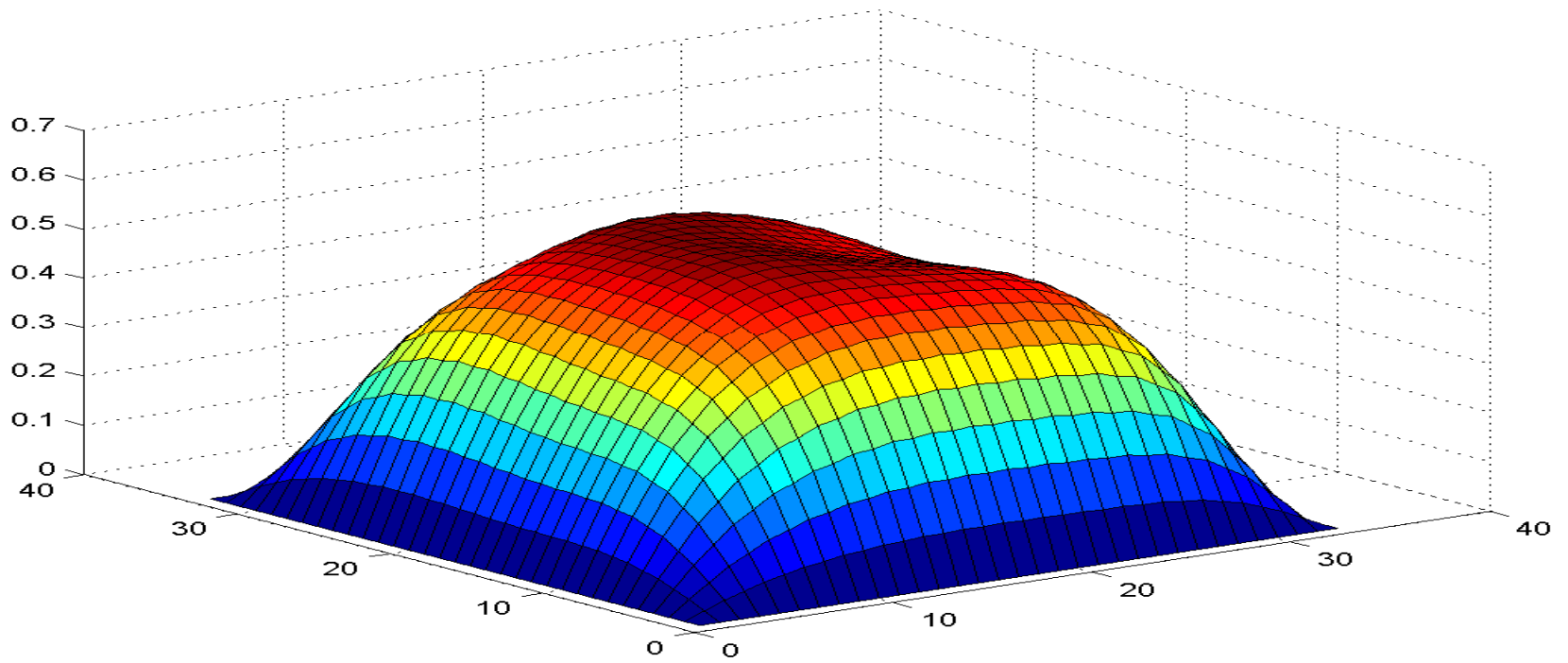


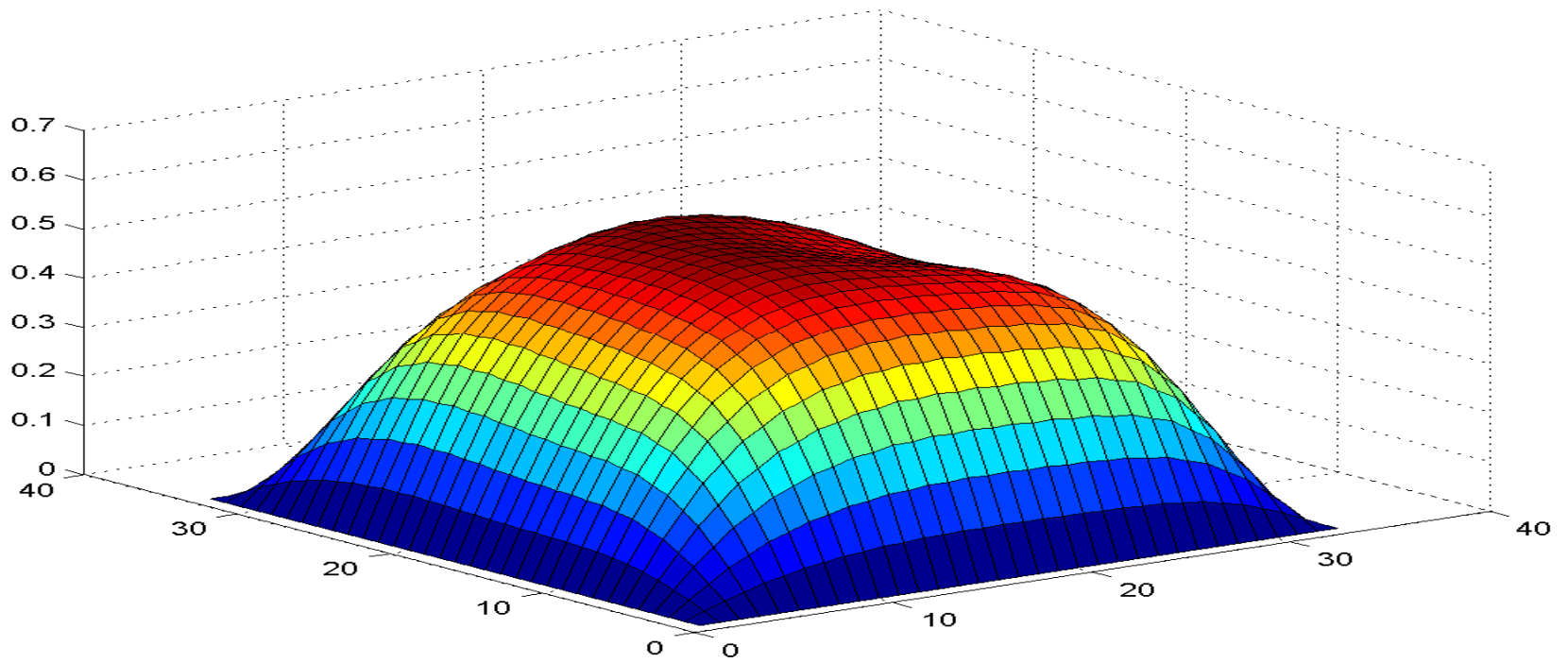


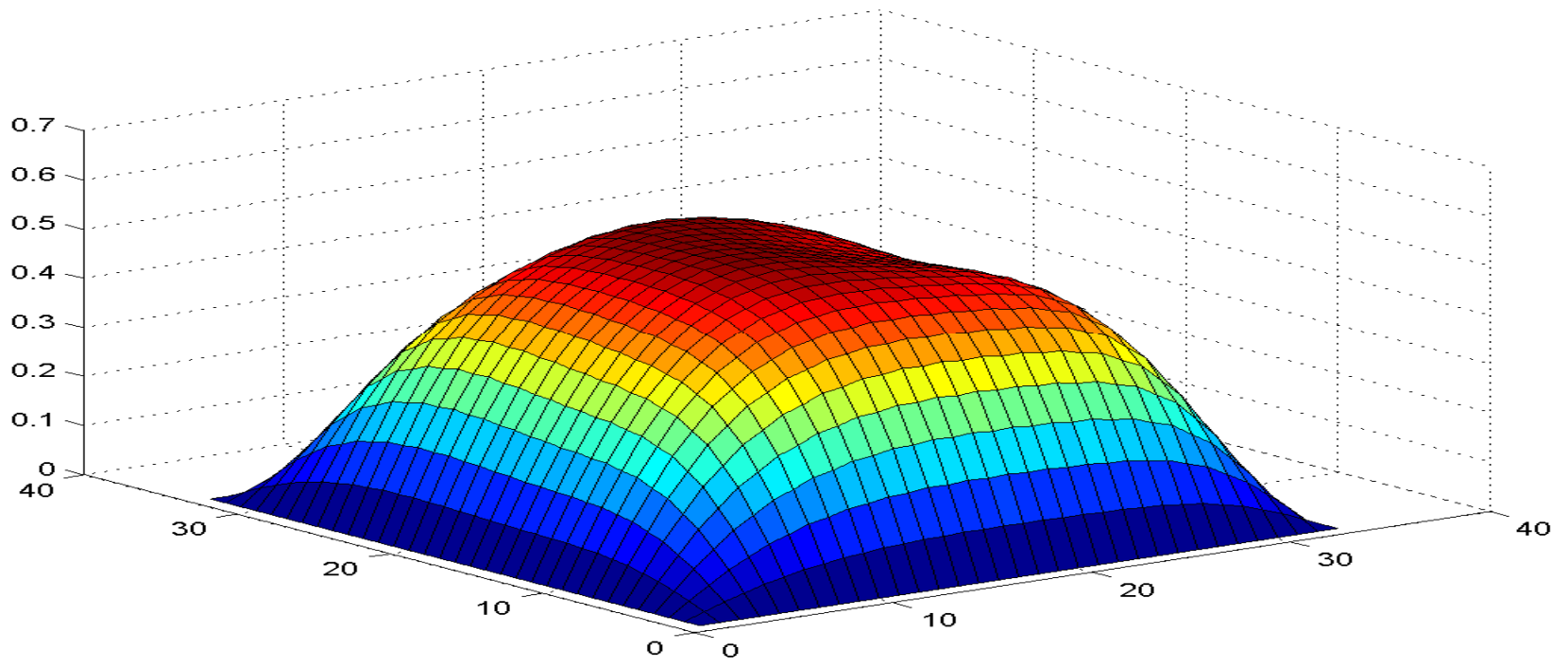


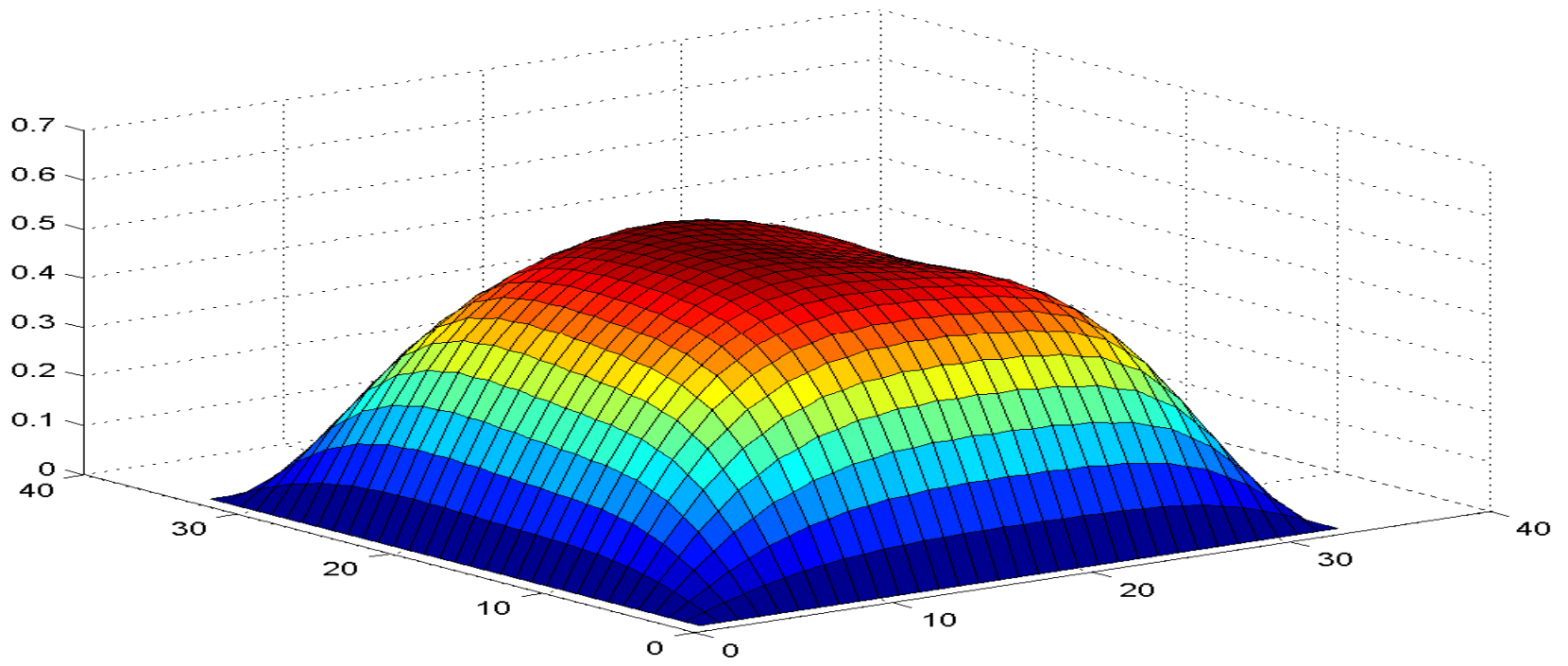


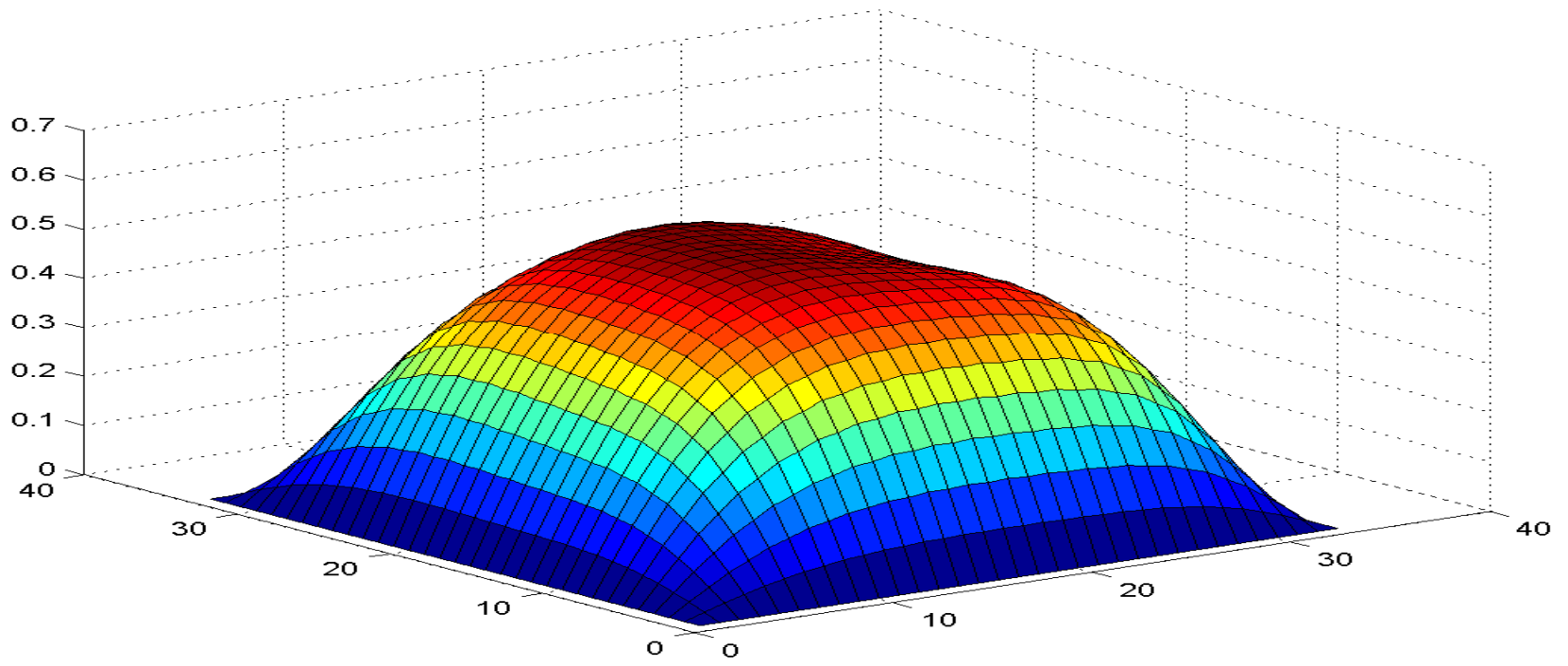












**Key Observation re-worded:** Relaxation cannot be generally efficient for reducing the error (i.e., the difference vector  $\tilde{u}^h - u^h$ ). But relaxation may be extremely efficient for **smoothing the error relative to the grid.**

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:

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

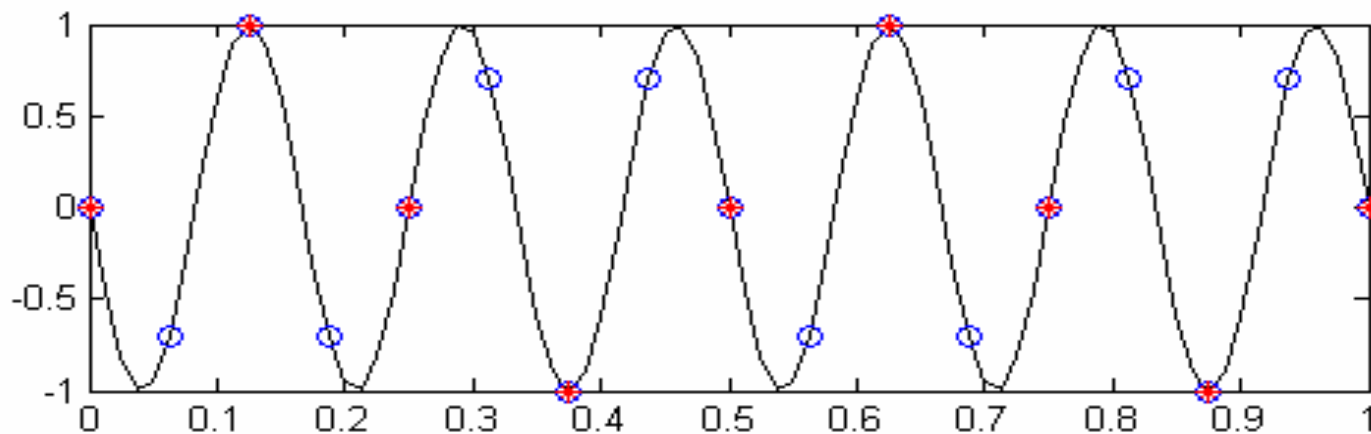
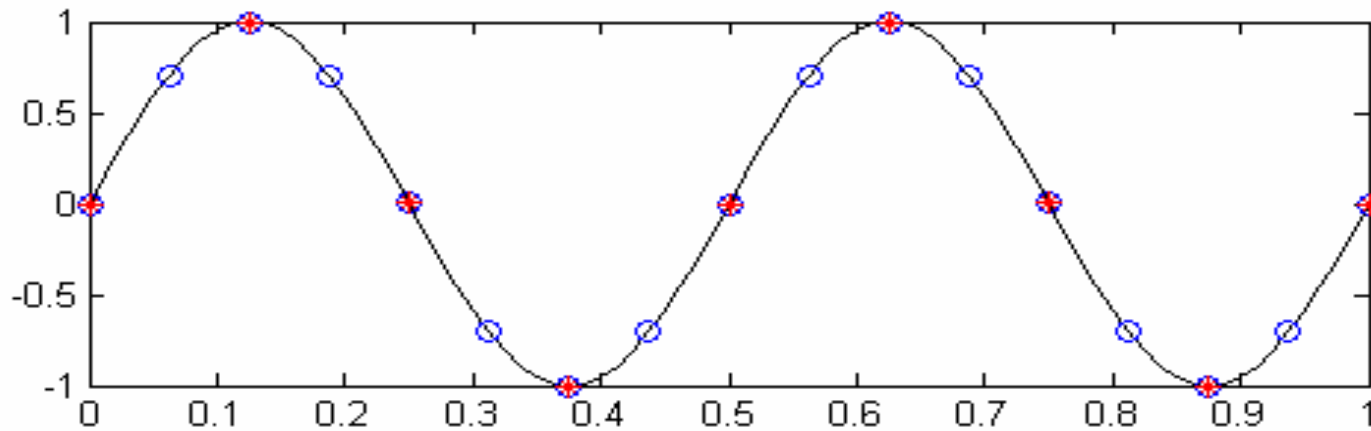
$$u_0^h = u_0,$$

$$u_N^h = u_1.$$

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

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





Aliasing

Smooth waves—with  $k \ll N$ —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:

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

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:  $I_h^H \cdot$
2. A **Prolongation** (coarse-to-fine) operator:  $I_H^h \cdot$

For restriction we can often use simple injection, but full-weighted transfers are preferable.

For prolongation linear interpolation (bi-linear in **2D**) is simple and usually effective.

## Two-grid 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 error-equation on the coarse grid:

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

- Interpolate and add correction:

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

- Relax again on grid  $h$ .

Multi-grid is obtained by recursion.

## Multi-grid Cycle $V(v_1, v_2)$

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

**Relax on**  $L^h u^h = f^h$   $v_1$  **times**

**Set**  $f^{2h} = I_h^{2h}(f^h - L^h u^h)$ ,  $u^{2h} = 0$

**Relax on**  $L^{2h} u^{2h} = f^{2h}$   $v_1$  **times**

**Set**  $f^{4h} = I_{2h}^{4h}(f^{2h} - L^{2h} u^{2h})$ ,  $u^{4h} = 0$

**Relax on**  $L^{4h} u^{4h} = f^{4h}$   $v_1$  **times**

**Set**  $f^{8h} = I_{4h}^{8h}(f^{4h} - L^{4h} u^{4h})$ ,  $u^{8h} = 0$

...

**Solve**  $L^{Mh} u^{Mh} = f^{Mh}$

...

**Correct**  $u^{4h} \leftarrow u^{4h} + I_{8h}^{4h} u^{8h}$

**Relax on**  $L^{4h} u^{4h} = f^{4h}$   $v_2$  **times**

**Correct**  $u^{2h} \leftarrow u^{2h} + I_{4h}^{2h} u^{4h}$

**Relax on**  $L^{2h} u^{2h} = f^{2h}$   $v_2$  **times**

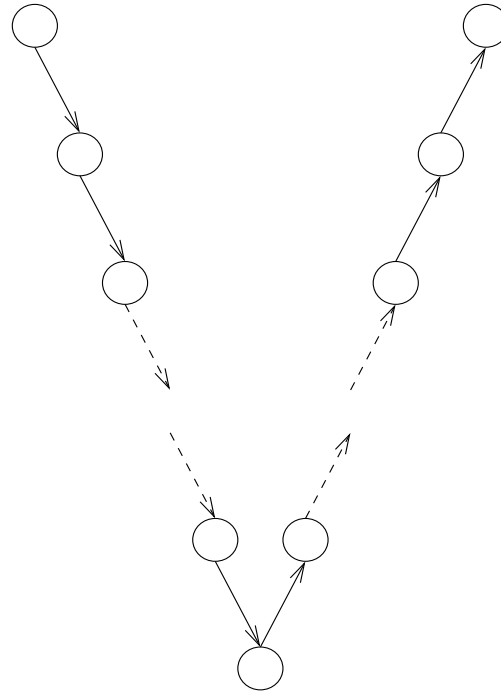
**Correct**  $u^h \leftarrow u^h + I_{2h}^h u^{2h}$

**Relax on**  $L^h u^h = f^h$   $v_2$  **times**

# V cycle

Finest grid

Coarsest grid



RELAXATION

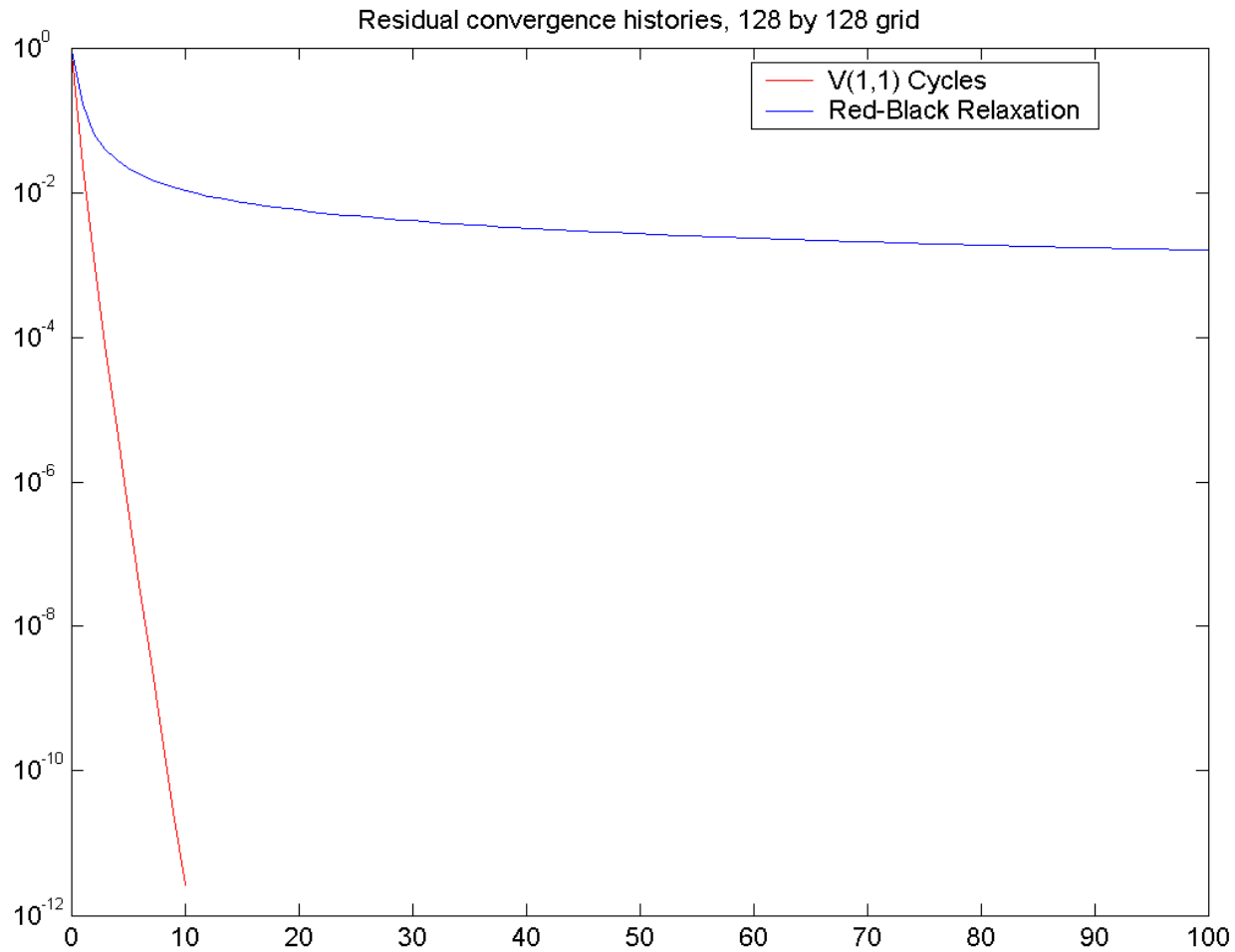


RESTRICTION



PROLONGATION





## Multigrid vs. Relaxation

## Remarks:

1. Simple recursion yields a  $V$  cycle. More generally, we can choose a **cycle index  $\gamma$** , and define a  $\gamma$ -cycle recursively as follows: Relax; transfer to next coarser grid; perform  $\gamma$   $\gamma$ -cycles; interpolate and correct; Relax. (On the coarsest grid define the  $\gamma$ -cycle as an exact solution).
2. The best number of pre-relaxation + post-relaxation sweeps is normally **2** or **3**.
3. The boundary conditions for all coarse-grid problems is zero (because the coarse-grid variable is the error). **The initial guess for the coarse-grid solution must be zero.**

## Local Mode Analysis (LMA)

We would like to obtain a quantitative prediction of the convergence behavior of the multigrid (or at least two-grid) cycle.

This is important for debugging, choosing parameters, etc.

We first derive the **iteration matrix** of the two-grid cycle. That is, the matrix  $\mathbf{S}^h$ ,

$$\mathbf{v}_{after}^h = \mathbf{S}^h \mathbf{v}_{before}^h ,$$

where  $\mathbf{v}_{before}^h$  and  $\mathbf{v}_{after}^h$  are the algebraic errors before and after the two-grid cycle.

## Notation

$\mathbf{R}^h$  – Relaxation matrix

$\mathbf{L}^h$  – Fine-grid matrix,  $\mathbf{L}^H$  – Coarse-grid matrix

$\mathbf{I}_H^h$  – Prolongation matrix,  $\mathbf{I}_h^H$  – Restriction matrix

$\mathbf{I}^h$  – Fine-grid identity matrix

$\nu_1$  – # Relaxations before CGC

$\nu_2$  – # Relaxations after CGC

## Two-grid matrix

$$\mathbf{S}^h = \left( \mathbf{R}^h \right)^{\nu_2} \left( \mathbf{I}^h - \mathbf{I}_H^h \left( \mathbf{L}^H \right)^{-1} \mathbf{I}_h^H \mathbf{L}^h \right) \left( \mathbf{R}^h \right)^{\nu_1}$$

For a given problem, we can compute the norm of  $S^h$  and determine the convergence behavior of the two-grid algorithm, which often provides a relevant approximation of the multigrid performance.

However, this requires use of a computer, and it is only moderately useful for algorithm development.

We can in fact obtain a useful quantitative approximate prediction by means of a local (Fourier) analysis.

## Fourier Analysis

Consider for simplicity the **1D** problem.

Instead of fixed boundary conditions we assume **periodicity**. Also, we assume our operator,  $\mathbf{L}^h$ , to have constant coefficients. Hence, every element of the corresponding matrix, denoted by  $\mathbf{A}$ , satisfies:

$$\mathbf{A}_{i,j} = \mathbf{A}_{i-1, j-1 \pmod{N}}.$$

That is, every row of  $\mathbf{A}$  is identical to the previous row, modulo  $N$ , shifted one place forward.

We next compute the **eigenvectors** and **eigenvalues** of matrices of this type.

Observation: any matrix  $\mathbf{A}$  representing a constant-coefficient discretization + periodicity can be written as a polynomial in the cyclic forward shift matrix,

$$\omega = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ & & & \ddots & & \\ 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & 0 \end{pmatrix} .$$

Here,

$$\omega_{i,j} = \delta_{i+1,j(\text{mod } N)} .$$

## Proof.

By induction, for any integer  $p$ ,

$$\left(\omega^p\right)_{i,j} = \delta_{i+p,j \pmod{N}}.$$

That is,  $\omega^p$  is a cyclic forward shift by  $p$  places. Hence we have

$$A = \sum_{j=0}^{N-1} A_{0,j} \omega^j.$$



Observation: If  $P$  is a polynomial and  $\mathbf{B}$  is a square matrix with eigenvector  $\mathbf{v}$  and corresponding eigenvalue  $\lambda$ , then  $\mathbf{v}$  is also an eigenvector of  $P(\mathbf{B})$  with corresponding eigenvalue  $P(\lambda)$ .

Proof:

Suppose  $P(\mathbf{B}) = \sum_{i=0}^m c_i \mathbf{B}^i$ .

Observe that, by induction,  $\mathbf{B}^i \mathbf{v} = \lambda^i \mathbf{v}$ .

Hence,

$$P(\mathbf{B}) \mathbf{v} = \sum_{i=0}^m c_i \lambda^i \mathbf{v} = P(\lambda) \mathbf{v}$$

Conclusion: Since any matrix  $\mathbf{A}$  of the type we are considering is a polynomial in  $\omega$ , we only need to compute the eigenvalues and eigenvectors of  $\omega$ . The corresponding eigenvalues of  $\mathbf{A}$  will then be easy to compute.

Let  $\lambda_k$  denote an eigenvalue of  $\omega$ , and let  $\mathbf{v}^k$  denote the corresponding eigenvector. We have :

$$\omega \begin{pmatrix} \mathbf{v}_0^k \\ \mathbf{v}_1^k \\ \vdots \\ \mathbf{v}_j^k \\ \vdots \\ \mathbf{v}_{N-2}^k \\ \mathbf{v}_{N-1}^k \end{pmatrix} = \begin{pmatrix} \mathbf{v}_1^k \\ \mathbf{v}_2^k \\ \vdots \\ \mathbf{v}_{j+1}^k \\ \vdots \\ \mathbf{v}_{N-1}^k \\ \mathbf{v}_0^k \end{pmatrix} = \lambda_k \begin{pmatrix} \mathbf{v}_0^k \\ \mathbf{v}_1^k \\ \vdots \\ \mathbf{v}_j^k \\ \vdots \\ \mathbf{v}_{N-2}^k \\ \mathbf{v}_{N-1}^k \end{pmatrix}$$

Hence,

$$\mathbf{v}_1^k = \lambda_k \mathbf{v}_0^k,$$

$$\mathbf{v}_2^k = \lambda_k \mathbf{v}_1^k = \lambda_k^2 \mathbf{v}_0^k,$$

...

$$\mathbf{v}_j^k = \lambda_k^j \mathbf{v}_0^k,$$

...

$$\mathbf{v}_0^k = \lambda_k \mathbf{v}_{N-1}^k = \lambda_k^N \mathbf{v}_0^k.$$

Thus, the  $N$  eigenvalues are the  $N$ th roots of 1:

$$\lambda_k = 1^{1/N} = e^{i2\pi k / N}.$$

The  $N$  eigenvectors are

$$\mathbf{v}^k = \begin{pmatrix} 1 \\ \lambda_k \\ \lambda_k^2 \\ \vdots \\ \lambda_k^j \\ \vdots \\ \lambda_k^{N-1} \end{pmatrix} = \begin{pmatrix} 1 \\ e^{ik2\pi/N} \\ e^{2ik2\pi/N} \\ \vdots \\ e^{jik2\pi/N} \\ \vdots \\ e^{(N-1)ik2\pi/N} \end{pmatrix} .$$

Here,  $k$  is any integer. But note that if  $k_1 = k_2 \pmod{N}$  then the eigenvalues corresponding to  $k_1$  and  $k_2$  alias. Thus,  $k$  runs over any  $N$  consecutive integers.

Consider now a grid of  $N$  equally spaced points over a domain of size  $2\pi$ . The grid-points are located at

$$x_j = hj = \frac{2\pi j}{N}, \quad j = 0, \dots, N-1,$$

where  $h = 2\pi / N$  is the mesh-size.

The eigenvectors can be considered as grid-based functions of the form:

$$\varphi_k(x_j) = e^{ikx_j}.$$

Here,  $k$  can be any integer, but note that  $\varphi_k$  aliases with  $\varphi_{k+N}$ . We therefore restrict:

$$k \in \left[ -\frac{N}{2} + 1, \frac{N}{2} \right].$$

The functions  $\varphi_k$  are called **Fourier components**. They possess some useful properties. Amongst these:

Define an inner product,

$$(f, g) = \frac{1}{N} \sum_{j=0}^{N-1} f(x_j) \bar{g}(x_j),$$

where "bar" denotes complex conjugate. Then the Fourier components are **orthogonal**:

$$(\varphi_k, \varphi_l) = \delta_{k, l(\text{mod } N)}.$$

The Fourier expansion is unique:

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

where the coefficients  $c_k$  are uniquely determined by

$$c_k = (f, \varphi_k).$$

Furthemore,

$$\sum |c_k|^2 = (f, f).$$

(Parseval's theorem).



## Additional Remarks.

1. Domain length  $2L$ : replace  $x$  by  $\pi x/L$ .
2. Nonperiodic, with function vanishing at endpoints: Antisymmetric continuation (sine series).
3. The wavelength is  $l=2\pi/k$ .
4. For multigrid analysis we define

$$\theta = hk = \frac{2\pi k}{N}, \quad -\pi < \theta \leq \pi.$$

Then,

$$\varphi_\theta(x_j) = e^{i\theta x_j/h} = e^{i\theta j}.$$

## Fourier Analysis

Local mode (Fourier) analysis is the main tool used for practical analysis of multigrid solvers. Though it is rigorously justified only for rather special situations, it is useful for quantitative predictions in a wide set of circumstances.

The underlying assumption is that small subsets comprised of one or a few Fourier components of the form (in  $d$  dimensions)

$$\varphi = e^{i\theta \cdot \mathbf{j}},$$

$$\theta = (\theta_1, \theta_2, \dots, \theta_d),$$

$$\mathbf{j} = \left( \frac{x_1}{h_1}, \frac{x_2}{h_2}, \dots, \frac{x_d}{h_d} \right),$$

are invariant under operations of the common multigrid components.

## The Symbol

The **symbol** is a generalization of the **eigenvalue**.

The symbol of an operator  $L$  is denoted by  $\hat{L}$ .

When the Fourier mode is an eigenfunction, it is defined by:

$$L e^{i\theta \cdot \mathbf{j}} = \hat{L}(\theta) e^{i\theta \cdot \mathbf{j}}.$$

## Examples (1D):

$$Lu = u_{xx},$$

$$L e^{i\theta x/h} = -\frac{\theta^2}{h^2} e^{i\theta x/h},$$

$$\Rightarrow \hat{L} = -\frac{\theta^2}{h^2}$$

Suppose we discretize  $L$  by

$$L^h u^h = \frac{u_{j+1}^h - 2u_j^h + u_{j-1}^h}{h^2}.$$

Then,

$$L^h e^{i\theta x/h} = \frac{e^{i\theta} - 2 + e^{-i\theta}}{h^2} e^{i\theta x/h}.$$

So,

$$\hat{L}^h(\theta) = \frac{2}{h^2} (\cos(\theta) - 1) = -\frac{4}{h^2} \sin^2\left(\frac{\theta}{2}\right).$$

Truncation error:

$$\hat{L}(\theta) - \hat{L}^h(\theta) = -\frac{\theta^2}{h^2} + \frac{4}{h^2} \sin^2\left(\frac{\theta}{2}\right) = O\left(\frac{\theta^4}{h^2}\right).$$

## The symbol of relaxation

Consider the discretized equation

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

A pointwise relaxation can often be written as

$$L^{h+} u_{after}^h + L^{h-} u_{before}^h = f^h ,$$

where  $u_{before}^h$  denotes the old approximation to  $u^h$  (before the relaxation step), and  $u_{after}^h$  denotes the new approximation.

Thus, the relaxation is characterized by the splitting:  $L^h = L^{h+} + L^{h-}$ .

## Examples:

For 1D Poisson,

$$L^h u^h = \frac{u_{j+1}^h - 2u_j^h + u_{j-1}^h}{h^2},$$

we obtain for Jacobi relaxation,

$$\left(L^{h+} u_{after}^h\right)_j = \frac{-2\left(u_{after}^h\right)_j}{h^2}, \quad \left(L^{h+} u_{before}^h\right)_j = \frac{\left(u_{before}^h\right)_{j+1} + \left(u_{before}^h\right)_{j-1}}{h^2},$$

and for Gauss-Seidel relaxation,

$$\left(L^{h+} u_{after}^h\right)_j = \frac{\left(u_{after}^h\right)_{j-1} - 2\left(u_{after}^h\right)_j}{h^2}, \quad \left(L^{h+} u_{before}^h\right)_j = \frac{\left(u_{before}^h\right)_{j+1}}{h^2},$$

Let  $v_{before}^h = u_{before}^h - u^h$  denote the algebraic error before the relaxation, and let  $v_{after}^h = u_{after}^h - u^h$  denote the new error. Then,

$$L^{h+} v_{after}^h + L^{h-} v_{before}^h = 0.$$

Now, consider an error that is a single Fourier component (assumed to be an eigenfunction of  $L^{h\pm}$ ):

$$v_{before}^h = A_{\theta} e^{i\theta x/h}, \quad v_{after}^h = \bar{A}_{\theta} e^{i\theta x/h}.$$

The relaxation operator,  $R^h$ , is defined by

$$v_{after}^h = R^h v_{before}^h.$$

We obtain that the symbol of  $R^h$  is:

$$\hat{R}^h(\theta) = \frac{\bar{A}_\theta}{A_\theta} = -\frac{\hat{L}^{h-}(\theta)}{\hat{L}^{h+}(\theta)}.$$

Examples: for Jacobi relaxation we have

$$\hat{L}^{h+}(\theta) = \frac{-2}{h^2}, \quad \hat{L}^{h-}(\theta) = \frac{e^{i\theta} + e^{-i\theta}}{h^2},$$

So the symbol of Jacobi relaxation is

$$\hat{R}_{Jac}^h(\theta) = \frac{e^{i\theta} + e^{-i\theta}}{2} = \cos(\theta).$$



For damped Jacobi relaxation we have

$$\begin{aligned}\bar{A}_\theta &= \left[ 1 - \omega + \omega \hat{R}_{Jac}^h(\theta) \right] A_\theta \\ &= \left[ 1 - \omega + \omega \cos(\theta) \right] A_\theta.\end{aligned}$$

So the symbol of damped Jacobi relaxation is

$$\hat{R}_{\omega Jac}^h(\theta) = 1 - \omega + \omega \cos(\theta).$$

For Gauss-Seidel relaxation we have

$$\left(v_{before}^h\right)_{j+1} - 2\left(v_{after}^h\right)_j + \left(v_{after}^h\right)_{j-1} = 0.$$

Substituting the Fourier function yields

$$\begin{aligned} A_\theta e^{i\theta(j+1)} - 2\bar{A}_\theta e^{i\theta j} + \bar{A}_\theta e^{i\theta(j-1)} &= \\ \left[ A_\theta e^{i\theta} - 2\bar{A}_\theta + \bar{A}_\theta e^{-i\theta} \right] e^{i\theta j} &= 0. \end{aligned}$$

Hence,

$$\bar{A}_\theta = \frac{e^{i\theta}}{2 - e^{-i\theta}} A_\theta.$$

The symbol of Gauss-Seidel relaxation is therefore

$$\hat{R}_{GS}^h(\theta) = \frac{e^{i\theta}}{2 - e^{-i\theta}}.$$

## Aliasing Revisited

Note that

$$\begin{aligned} e^{i(\theta \pm 2\pi)x_j/h} &= e^{i(\theta \pm 2\pi)j} = \\ e^{\pm i2\pi j} e^{i\theta j} &= e^{i\theta j} = e^{i\theta x_j/h}. \end{aligned}$$

That is, Fourier modes  $e^{i\theta x/h}$  and  $e^{i(\theta \pm 2\pi)x/h}$  **alias** with each other on grid  $h$ .

On grid  $2h$  the component  $e^{i\theta x/h}$  becomes  $e^{i2\theta x/2h}$ .

That is, its frequency relative to the grid is **doubled**.

## Aliasing Revisited

Fourier modes  $e^{i\theta x/h}$  and  $e^{i(\theta \pm 2\pi)x/h}$  alias with each other on grid  $h$ .

On grid  $2h$  the component  $e^{i\theta x/h}$  becomes  $e^{i2\theta x/2h}$ .  
That is, its frequency relative to the grid is doubled.

Thus, the fine-grid components  $e^{i\theta x/h}$  and  $e^{i(\theta \pm \pi)x/h}$  alias when sampled on grid  $2h$ .

**Conclusion:** the coarse grid  $2h$  resolves only about  $1/2$  of the fine-grid frequencies - those in the range  $|\theta| \leq \pi / 2$ .

## Smoothing analysis

We simplify the analysis of the two-grid algorithm by making the following approximation:

1. The coarse-grid correction **eliminates all smooth** fine-grid components, those with  $|\theta| \leq \pi / 2$  .
2. The coarse-grid correction has **no effect on the rough** fine-grid error components, with  $|\theta| > \pi / 2$  .

With these simplifications we can predict approximately the **convergence rate per fine-grid relaxation sweep** of the two-grid cycle by computing the **smoothing factor** defined below.

## The smoothing factor

Let  $\hat{R}^h$  denote the symbol of a relaxation operator whose eigenvectors are Fourier components. Then the **smoothing factor** is defined by

$$\mu = \max_{\pi/2 \leq |\theta| \leq \pi} \left| \hat{R}^h(\theta) \right|.$$

Example:

The smoothing factor of Gauss-Seidel relaxation is

$$\begin{aligned}\mu &= \max_{\pi/2 \leq |\theta| \leq \pi} \left| \hat{R}_{GS}^h(\theta) \right| \\ &= \max_{\pi/2 \leq |\theta| \leq \pi} \left| \frac{e^{i\theta}}{2 - e^{-i\theta}} \right| \\ &= \max_{\pi/2 \leq |\theta| \leq \pi} \left| \frac{1}{2 - \cos(\theta) + i \sin(\theta)} \right| \\ &= \max_{\pi/2 \leq |\theta| \leq \pi} \left| \frac{1}{\sqrt{(2 - \cos(\theta))^2 + \sin^2(\theta)}} \right| = \frac{1}{\sqrt{5}}.\end{aligned}$$

The maximum is obtained at  $\theta = \pm\pi/2$ .

The smoothing factor of **Jacobi relaxation** with no damping is 1, because  $|\cos(\pm\pi)| = 1$ . Jacobi relaxation does not smooth errors at the highest frequencies, only changes their signs.

What is the smoothing factor of **damped Jacobi**? What is the **optimal damping**? That is, what is the  $\omega$  which minimizes  $\mu$  of damped Jacobi? What is the corresponding  $\mu$ ?



## Higher dimensions

The Fourier analysis can be generalized to  $d$  dimensions.

$$\mathbf{x}_j = \left( x_{j_1}^{(1)}, x_{j_2}^{(2)}, \dots, x_{j_d}^{(d)} \right), \quad \theta = (\theta_1, \theta_2, \dots, \theta_d),$$

where

$$x_{j_k}^{(k)} = j_k h_k, \quad k = 1, \dots, d.$$

## Higher dimensions

The Fourier components are

$$\varphi_{\theta}(\mathbf{x}_j) = e^{i\left(\sum_{k=1}^d \theta_k x_{j_k}^{(k)} / h_k\right)} = e^{i\left(\sum_{k=1}^d \theta_k j_k\right)},$$

Where  $h_k$  is the mesh-size in the  $k$ th coordinate.

The **smoothing factor** (for standard coarsening) is now defined as in the **1D** case, with

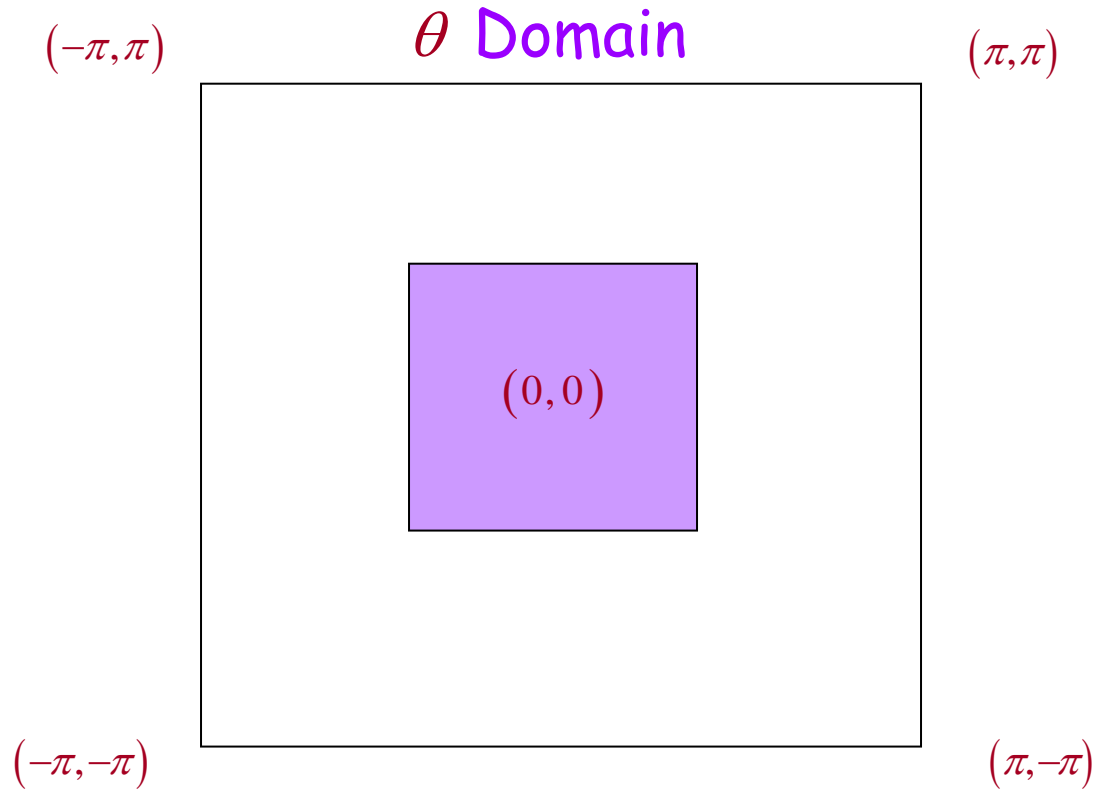
$$|\theta| \stackrel{def}{=} \max (|\theta_1|, \dots, |\theta_d|).$$

### Example (2D)

The **symbol of Gauss-Seidel Relaxation** for the **Poisson problem** is

$$\hat{R}(\theta) = \frac{e^{i\theta_1} + e^{i\theta_2}}{4 - e^{-i\theta_1} - e^{-i\theta_2}}.$$

The **smoothing factor**  $\mu$  is the maximum of  $|\hat{R}(\theta)|$  over all  $\theta$  for which the absolute value of at least one component is at least  $\pi/2$ .



The **shaded area** marks the part of the  $\theta$  domain that is ignored when computing the **smoothing factor**.

The **smoothing factor** is most easily computed approximately by a small computer program. For **Gauss-Seidel relaxation** of the **5-point Laplacian**, the smoothing factor is found to be  $\frac{1}{2}$ .

**Conclusion:** a  **$V(2,1)$  cycle** is expected to reduce the error approximately by a factor **8** per cycle.

## Exercise:

Write a MATLAB function that computes **approximately** the smoothing factor,  $\mu$ . The input should be two small matrices representing the stencils of  $L^{h+}$  and  $L^{h-}$ , and also a damping parameter,  $\omega$ . The function should compute the **symbol** of the relaxation, and maximize its absolute value over a discrete subset spanning the high frequencies,

$$(\theta_1, \theta_2) \subset [-\pi, \pi] \times [-\pi, \pi] \setminus \left[ -\frac{\pi}{2}, \frac{\pi}{2} \right] \times \left[ -\frac{\pi}{2}, \frac{\pi}{2} \right].$$

The resolution of the  $\theta$ 's should also be a parameter (run with 65 by 65). Use the program to verify the analytical results of the smoothing analysis for Jacobi relaxation (presented later). Verify also that the smoothing factor of point Gauss-Seidel for the 5-point Laplacian is 0.5.

For example, for Gauss-Seidel the input is

$$L^{h+} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & -4 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad L^{h-} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix},$$

whereas for Jacobi it is

$$L^{h+} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -4 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad L^{h-} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

The symbol of the relaxation is then given by

$$R^h = 1 - \omega + \omega \left( -\frac{\hat{L}^{h-}}{\hat{L}^{h+}} \right).$$

## Ellipticity and $h$ -Ellipticity

Multigrid methods are particularly useful and often straightforward for **elliptic problems**. But the “on-off” definition of ellipticity is inadequate for numerical purposes, and a **quantitative** measure of “ellipticity” of the discrete operator is important. This is given by the  $h$ -ellipticity measure,  $E^h$ , defined by

$$E^h(L^h) = \frac{\min_{\pi/2 \leq |\theta| \leq \pi} |\hat{L}^h(\theta)|}{\max_{\pi/2 \leq |\theta| \leq \pi} |\hat{L}^h(\theta)|},$$

where  $\hat{L}^h$  is the symbol of  $L^h$ .



We say that a discrete operator,  $L^h$ , is  $h$ -elliptic if  $E^h$  is bounded away from zero. Generally, for ordinary (i.e., local) relaxation methods, larger  $E^h$  corresponds to better "error-smoothability" by local processing.

Basically, a large  $E^h$  implies that all high-frequency errors generate relatively large residuals that are roughly of the same size. If  $E^h$  is small, then there are some error components whose residuals are relatively small. Such components cannot be detected locally, and hence they cannot be reduced efficiently by a local relaxation.

**Remark:** an elliptic PDO may have a discretization that is *not*  $h$ -elliptic, while a nonelliptic PDO might have one that *is*.

Examples:

Consider three different stencils for the Laplacian .

$$L^h = \frac{1}{h^2} \begin{bmatrix} \cdot & 1 & \cdot \\ 1 & -4 & 1 \\ \cdot & 1 & \cdot \end{bmatrix}. \quad (1)$$

Here,

$$\hat{L}^h(\theta) = \frac{1}{h^2} (2 \cos \theta_1 + 2 \cos \theta_2 - 4) = -\frac{4}{h^2} (\sin^2(\theta_1/2) + \sin^2(\theta_2/2)).$$

Hence,  $E^h = 1/4 = O(1)$ , and we say that  $L^h$  is  $h$ -elliptic.

Next, choose

$$L^h = \frac{1}{2h^2} \begin{bmatrix} 1 & \cdot & 1 \\ \cdot & -4 & \cdot \\ 1 & \cdot & 1 \end{bmatrix}. \quad (2)$$

Here,

$$\hat{L}^h(\theta) = \frac{2}{h^2} (\cos \theta_1 \cos \theta_2 - 1),$$

and

$$E^h = 0.$$

**Conclusion:** if this discretization is used, the error cannot be smoothed efficiently by local relaxation.

Finally, choose

$$L^h = \frac{1}{4h^2} \begin{bmatrix} \cdot & \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & \cdot & -4 & \cdot & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot & \cdot \end{bmatrix}. \quad (3)$$

Here,

$$\hat{L}^h(\theta) = \frac{1}{4h^2} (2 \cos 2\theta_1 + 2 \cos 2\theta_2 - 4) = -\frac{4}{h^2} (\sin^2 \theta_1 + \sin^2 \theta_2).$$

Again  $E^h = 0$ . Note that in the last two examples, the Red-Black mode could not be smoothed.

## Smoothing and $h$ -ellipticity

$h$ -ellipticity is a necessary and sufficient condition for the existence of pointwise smoothers based on a local splitting,

$$L^h = L^{h+} + L^{h-},$$

where  $L^{h+}$  is comprised of the coefficients of variables already relaxed, while  $L^{h-}$  is comprised of the coefficients of unrelaxed variables.

Obviously,  $E^h(L^h) = 0$  implies  $\hat{L}^h(\theta^*) = 0$  for some high frequency  $\theta^*$ , and therefore

$$L^{h+}(\theta^*) = -L^{h-}(\theta^*).$$

For  $\hat{L}^{h+}(\theta^*) \neq 0$ , we get

$$\left| \hat{R}(\theta^*) \right| = \left| \frac{\hat{L}^{h-}(\theta^*)}{\hat{L}^{h+}(\theta^*)} \right| = 1.$$

Thus the smoothing factor is at best 1.

## Exercise:

Show that **optimally damped Jacobi relaxation** yields for symmetric constant-coefficient operators, a smoothing factor

$$\mu = \frac{1 - E^h}{1 + E^h},$$

where  $E^h$  is the  $h$ -ellipticity measure. (Note that the symmetry of the operators implies that the **symbols are real**).

**Hint:** without loss of generality, assume that  $L^h$  is normalized such that its diagonal is the identity matrix. Then, write the damped Jacobi relaxation matrix in terms of  $L^h$ .

For **non-symmetric stencils** we can obtain the bound

$$\mu \leq \frac{1 - (E^h)^2}{1 + (E^h)^2} < 1.$$

By means of a **distributive relaxation**. However, in practice we can usually find far simpler and more effective smoothers.



## Anisotropic Diffusion

Let 
$$Lu = u_{ss} + \varepsilon u_{tt}.$$

Let  $\phi$  be the angle between  $(s,t)$  and the grid-aligned coordinate system,  $(x,y)$ . Hence,

$$Lu = (C^2 + \varepsilon S^2)u_{xx} + 2(1 - \varepsilon)CSu_{xy} + (\varepsilon C^2 + S^2)u_{yy},$$

with

$$C = \cos(\phi), \quad S = \sin \phi.$$

## Anisotropic Diffusion

We discretize  $L$  using the stencil

$$L^h = \frac{1}{h^2} \begin{bmatrix} -\frac{1}{2}(1-\varepsilon)CS & \varepsilon C^2 + S^2 & \frac{1}{2}(1-\varepsilon)CS \\ C^2 + \varepsilon S^2 & -2(1+\varepsilon) & C^2 + \varepsilon S^2 \\ \frac{1}{2}(1-\varepsilon)CS & \varepsilon C^2 + S^2 & -\frac{1}{2}(1-\varepsilon)CS \end{bmatrix}.$$

## Anisotropic Diffusion: The Aligned case

In the previous examples, of the Laplacian operator, we were able to find a discretization that would give us a good *h*-ellipticity measure. But suppose that the differential operator is

$$L = \varepsilon \partial_{xx} + \partial_{yy}$$

A 2<sup>nd</sup> order discretization for the aligned linear diffusion operator is again obtained by the standard five-point stencil:

$$L^h = \frac{1}{h^2} \begin{bmatrix} \cdot & 1 & \cdot \\ \varepsilon & -2(1 + \varepsilon) & \varepsilon \\ \cdot & 1 & \cdot \end{bmatrix} .$$

Here,

$$\begin{aligned} \hat{L}^h &= \frac{1}{h^2} \left[ 2 \left( \varepsilon \cos(\theta_1) + \cos(\theta_2) - 2(1 + \varepsilon) \right) \right] \\ &= -\frac{4}{h^2} \left[ \varepsilon \sin^2(\theta_1/2) + \sin^2(\theta_2/2) \right] . \end{aligned}$$

Setting  $(\theta_1, \theta_2) = (\pi/2, 0)$  vs.  $(\pi, \pi)$ , for example, we obtain  $E^h = O(\varepsilon)$ ,  $\varepsilon \rightarrow 0$ , hence small  $h$ -ellipticity. Indeed, all error components which have high-frequency oscillations only in the  $x$  direction generate relatively small residuals. Such errors cannot be reduced efficiently by local relaxation.

For example, the symbol of the Gauss-Seidel relaxation for this operator is

$$\hat{R}(\theta) = \frac{\varepsilon e^{i\theta_1} + e^{i\theta_2}}{2(1 + \varepsilon) - \varepsilon e^{-i\theta_1} - e^{-i\theta_2}} \quad (9)$$

setting  $\theta = (\pi, 0)$ , for example we get

$$|\hat{R}(\pi, 0)| = \frac{1 - \varepsilon}{1 + 3\varepsilon} = 1 - 4\varepsilon + O(\varepsilon^2), \quad \varepsilon \rightarrow 0 \quad (10)$$

Thus the smoothing factor is very poor for small  $\varepsilon$  as expected.

## Treating Aligned Anisotropy

There are two general approaches for handling anisotropic operators. One method is to employ **line-relaxation in the direction of the strong coupling** (i.e., for which the coefficient is relatively large - the  $y$  direction in this example.) This means that we **relax simultaneously a full line of variables** for each gridpoint index  $i$  in the  $x$  direction.

In our **Gauss-Seidel** example, the resulting **relaxation symbol** is

$$\hat{R}(\theta) = \frac{\varepsilon e^{i\theta_1}}{2(1 + \varepsilon) - \varepsilon e^{-i\theta_1} - e^{i\theta_2} - e^{-i\theta_2}}$$

Now,  $|R(\theta)|$  is maximized over the high frequencies for  $\theta = (\pi/2, 0)$ , yielding  $|R(\pi/2, 0)| = 1/\sqrt{5}$ , for  $\varepsilon \rightarrow 0$ , which implies very good smoothing.

The drawback of this approach is that for each  $y$ -line we must solve a tri-diagonal system of equations. We can do this by the usual Gaussian elimination or by a 1D multigrid cycle.

An alternative approach for anisotropic operators is **partial coarsening (or semi-coarsening)**. In this approach we use the usual relaxation, but we only **coarsen in the direction in which the error is smoothed efficiently** ( $y$  in our example).

Thus, the relaxation symbol remains as usual, but **the coarse grid resolves more components**, and the definition of the smoothing factor changes accordingly.



In our example, the **smoothing factor** is given by

$$\mu = \max_{\pi/2 \leq \theta_2 \leq \pi} \left| \hat{R}(\theta) \right|,$$

where  $\hat{R}(\theta)$  is the usual Gauss-Seidel symbol (9). Given the range of  $\theta$ , the maximum (for small  $\varepsilon$ ) is now obtained at  $\theta = (0, \pi/2)$ , yielding,

$$\left| \hat{R}(\theta) \right| = \left| \frac{i + \varepsilon}{2(1 + \varepsilon) - \varepsilon - i} \right| \xrightarrow{\varepsilon \rightarrow 0} \frac{1}{\sqrt{5}},$$

again implying good smoothing properties.

## The General Rule of Block Relaxation

The general rule is that local (point) relaxation **only smoothes the error efficiently in the direction of the strongest coupling**. If we relax the strongly-coupled variables simultaneously (block relaxation), then the relaxation will smooth well also in the direction of the second-strongest coupling. Thus, we can use block-relaxation (line, plane, etc.) to **regain full multigrid efficiency**.

Alternatively, we can refrain from coarsening in the directions along which the error is not smoothed. **The ultimate form of this is Algebraic Multigrid (AMG)**.

Both techniques can be used simultaneously. For example, if we do not know *a priori* the direction of strong coupling, then we can use line relaxation in one direction while coarsening only in the other. This can be generalized to higher dimensions.

Alternatively, we can use line relaxation in each of the directions (alternating), but then the generalization to higher dimensions is more cumbersome.

## Conclusions

- With proper care and insight, multigrid methods are a highly efficient tool for the iterative solution of problems arising from the discretization of elliptic PDE
- Honing your multigrid algorithm to make it work for your particular problem may be difficult.
- A more general and robust approach - that comes with the cost of heavier machinery - is Algebraic Multigrid Methods, introduced in the next tutorial.