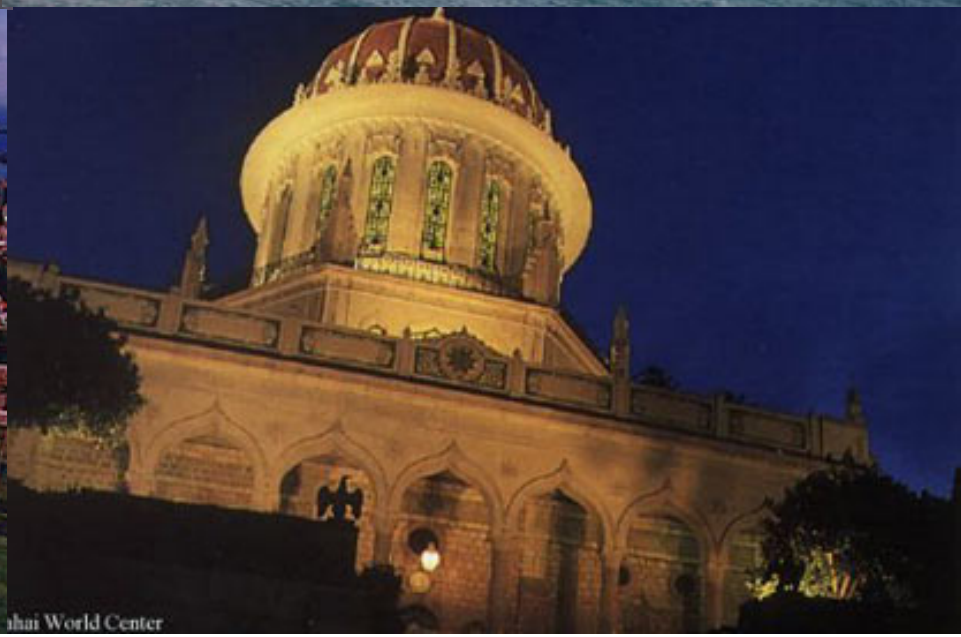
A Multigrid Tutorial

Irad Yavneh

Faculty of Computer Science

Technion – Israel Institute of Technology

irad@cs.technion.ac.il



hai World Center

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.

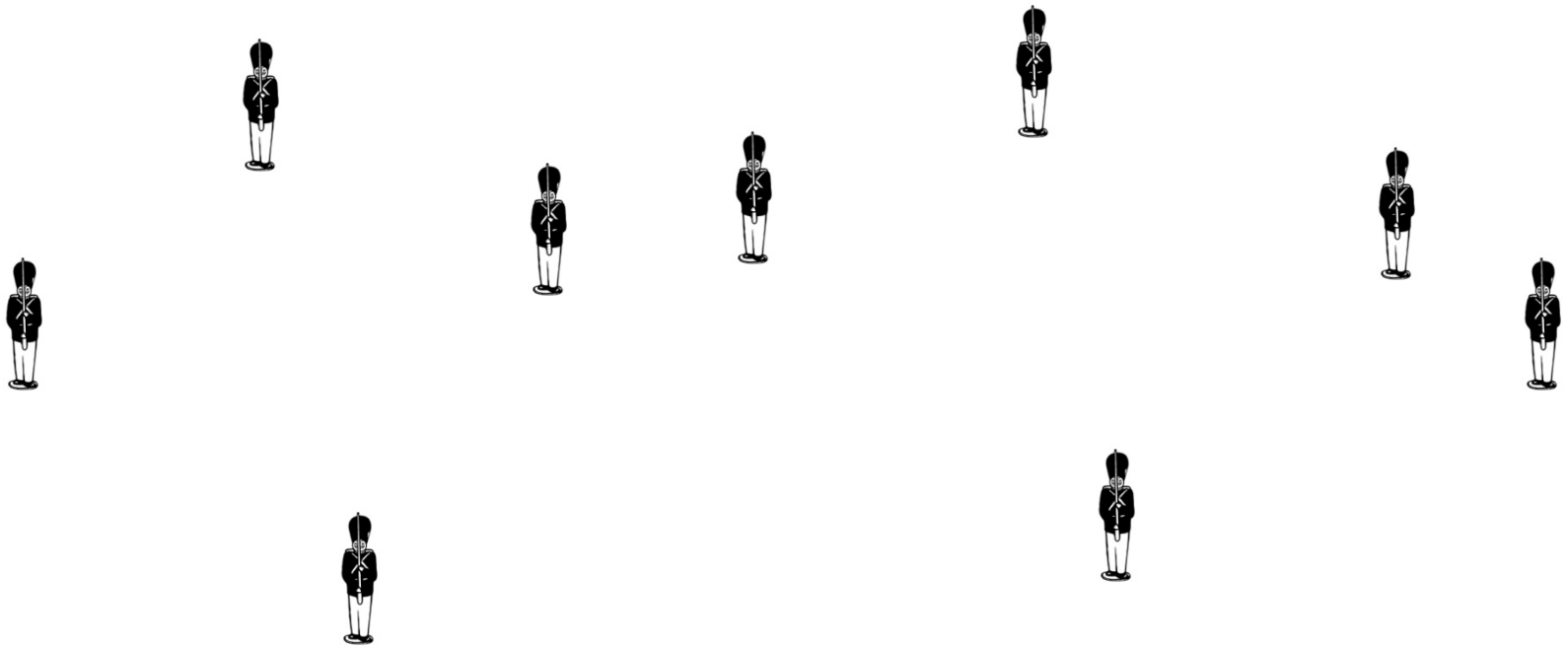
What's it about?

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

Basic Concepts: Local vs. Global processing.

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

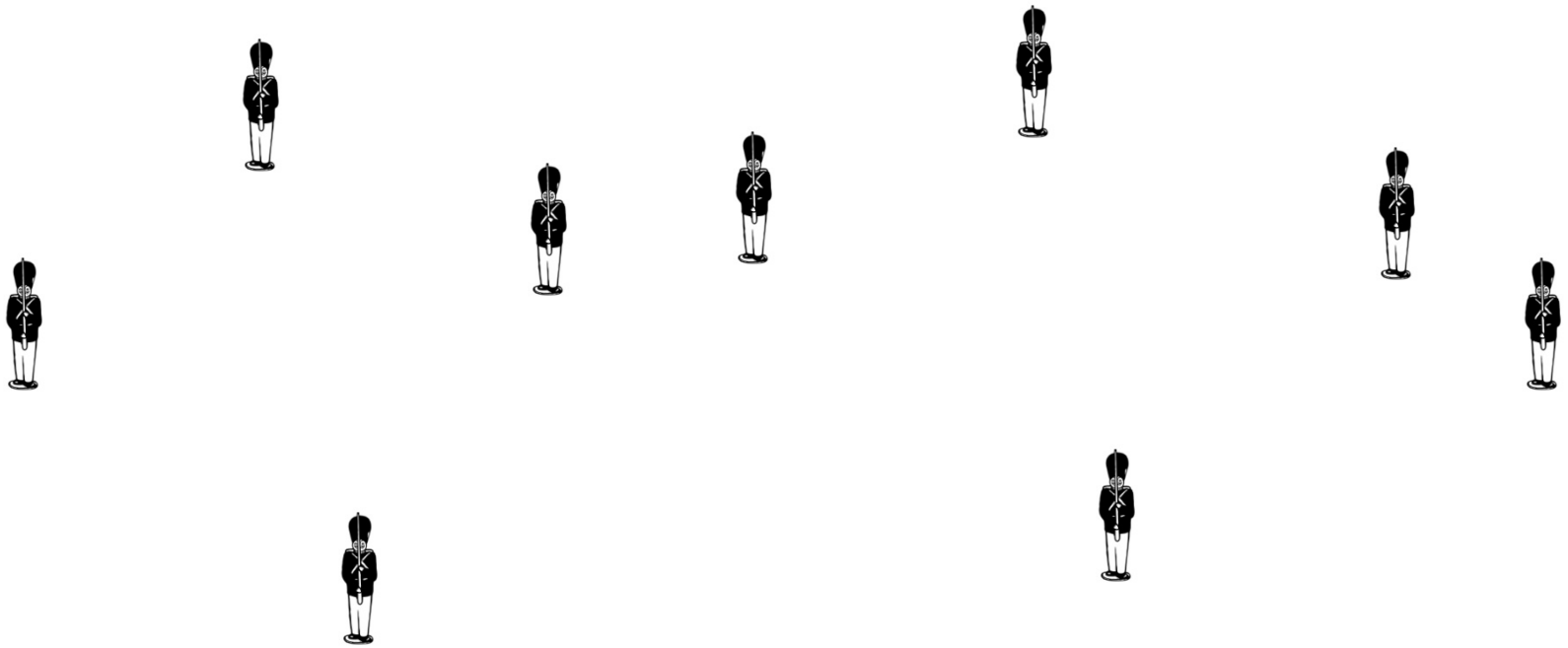
The two soldiers at the ends of the line are fixed. Suppose we number the soldiers 0 to N , and that the length of the entire line is L.



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.

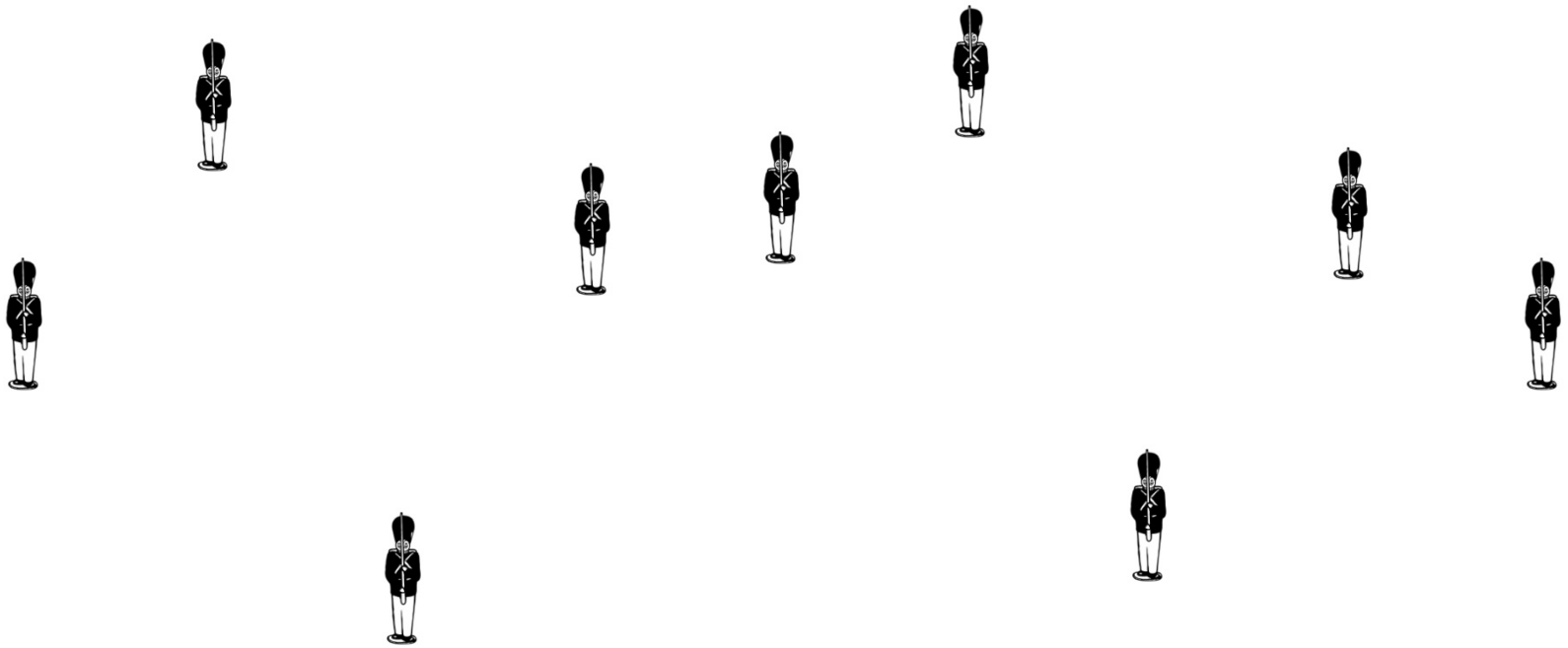


This method solves the problem directly, but it requires a high degree of sophistication: recognition of the extreme soldiers and some pretty fancy arithmetic.

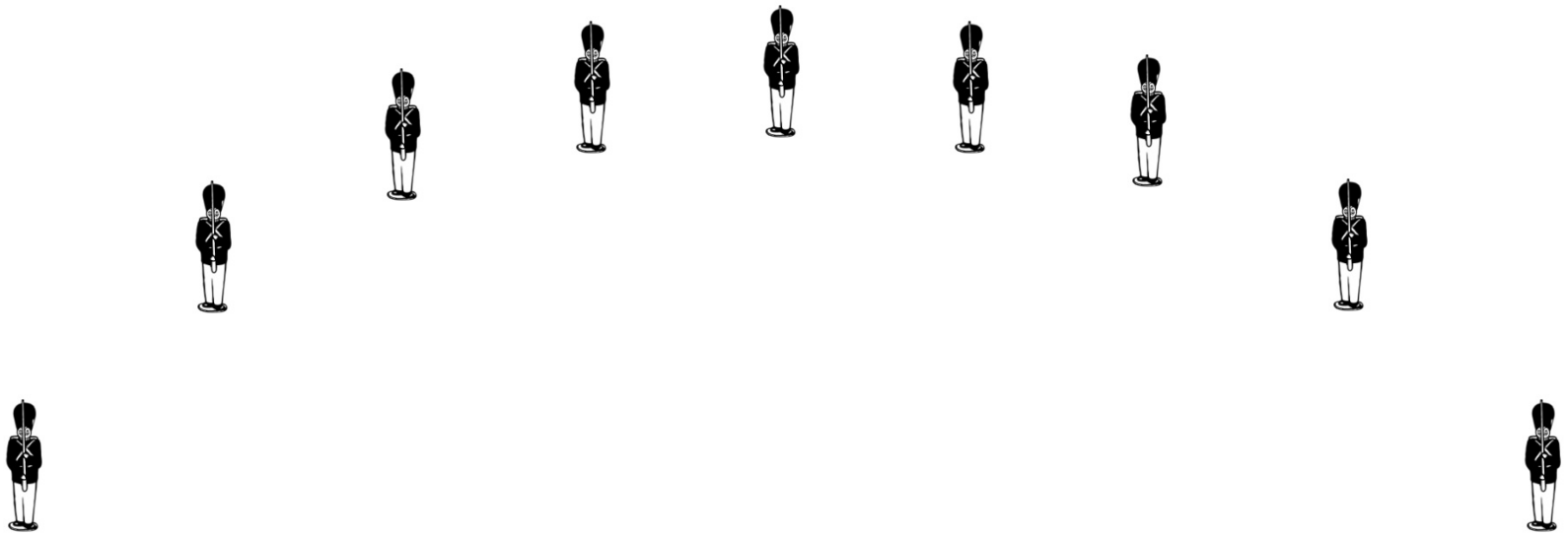
Local processing (iterative method). Suppose that the inner soldiers' initial position is $\mathbf{x}^{(0)} = (x_1, x_2, \ldots, x_{N-1})^{(0)}$. Then repeat for i=1,2,…: Let each soldier j, j=1,…N-1 at iteration i move to the point midway between the locations of soldier j-1 and soldier j+1 at iteration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

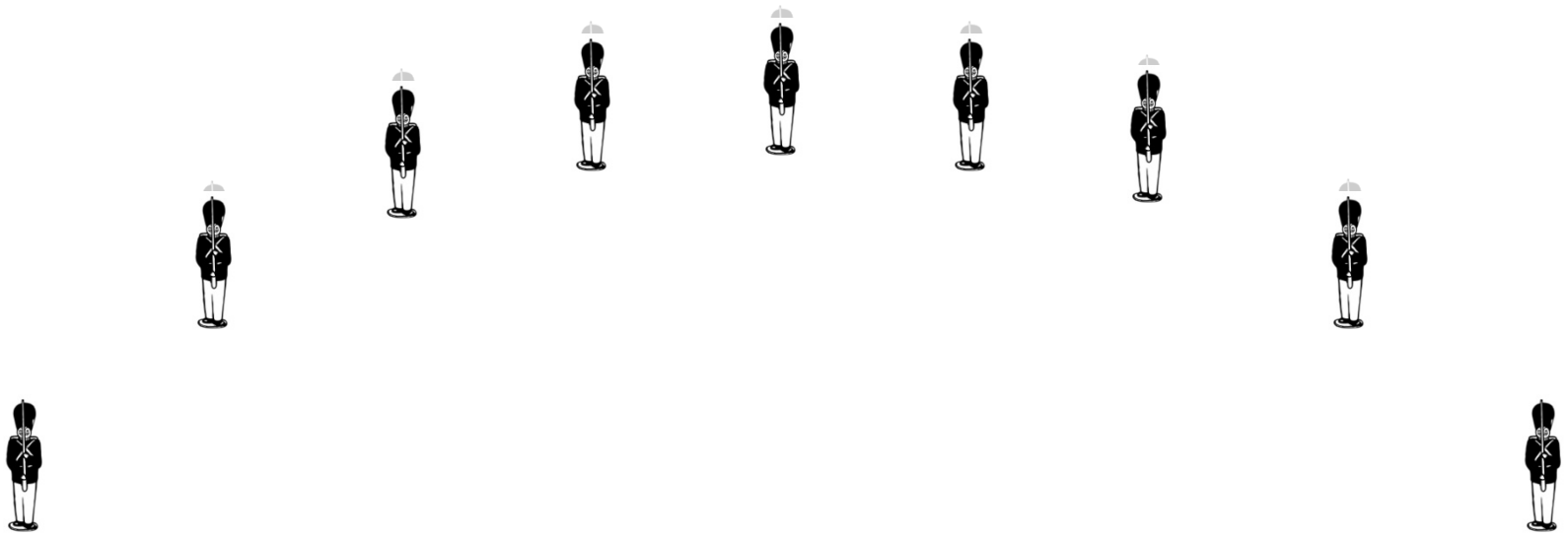
$$x_j^{(i)} = \frac{1}{2}\left(x_{j-1}^{(i-1)} + x_{j+1}^{(i-1)}\right)$$

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

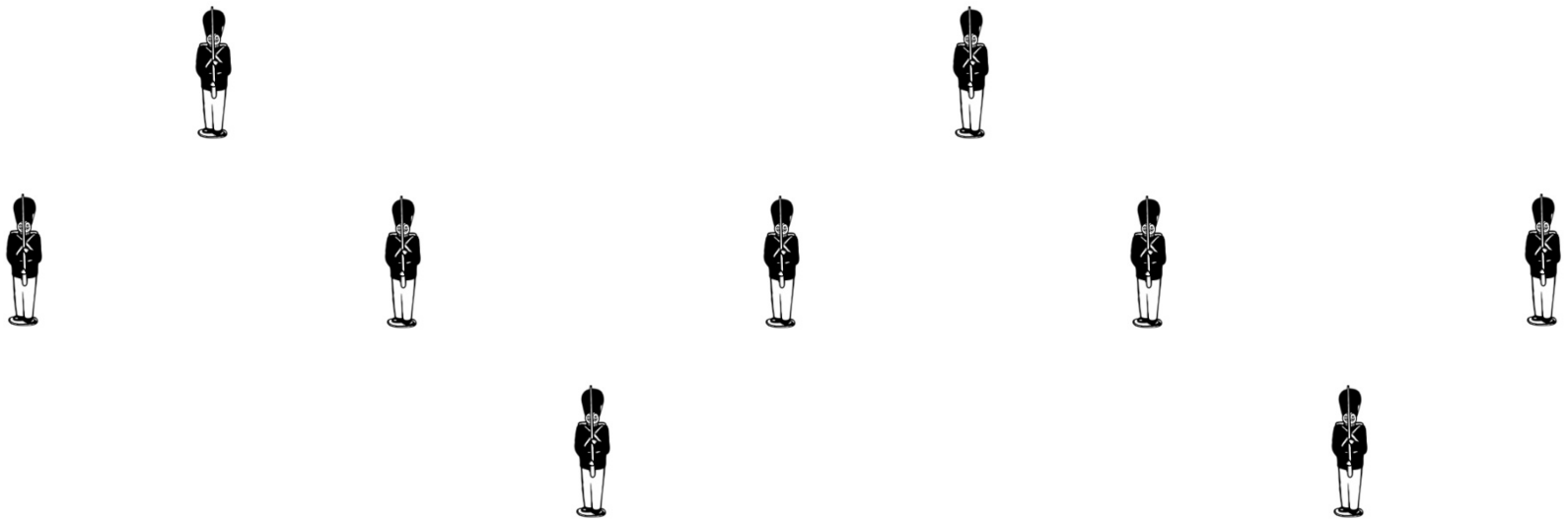


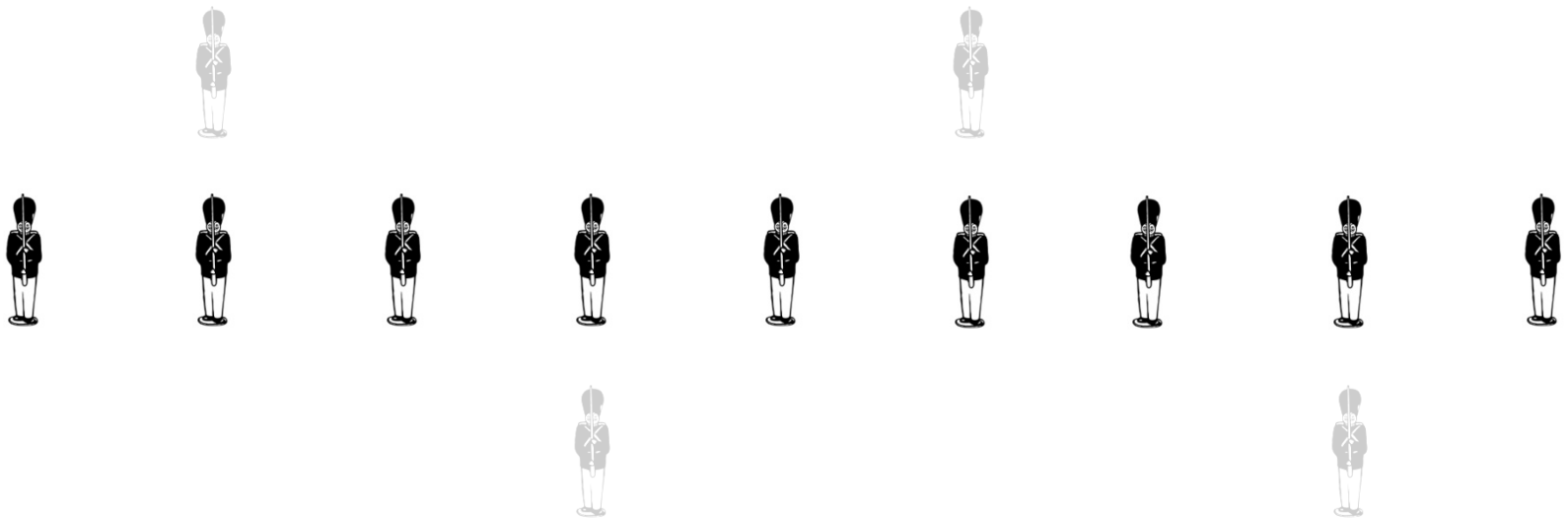
A step in the right direction





Slow convergence





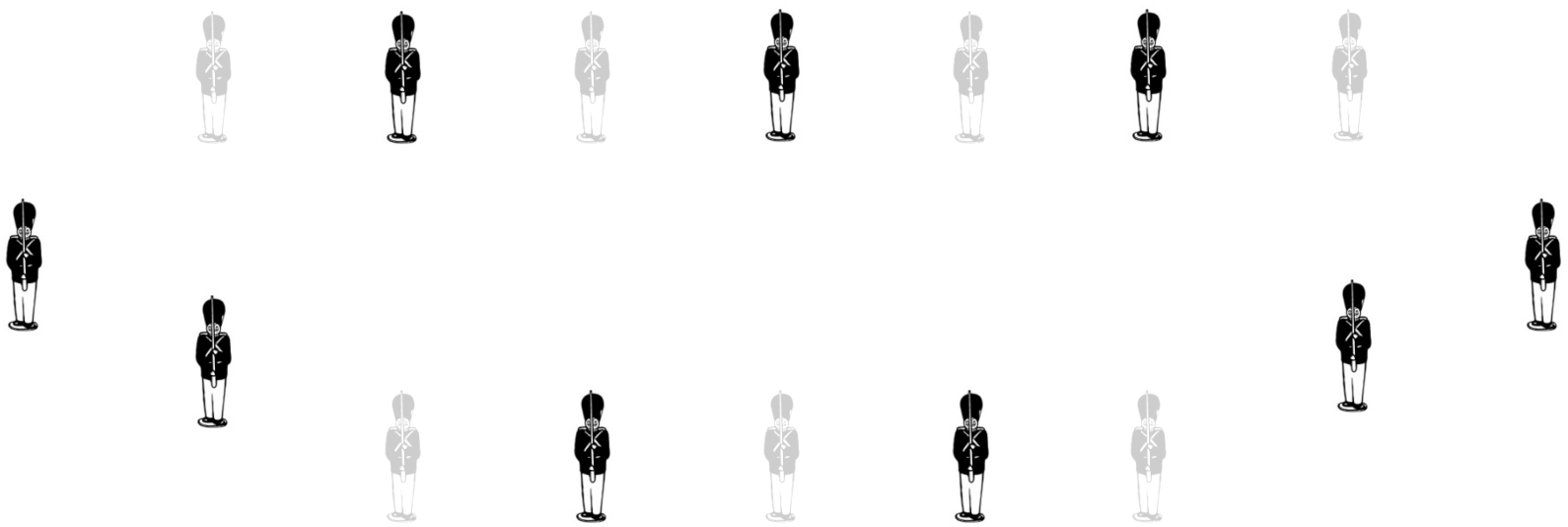
Fast convergence



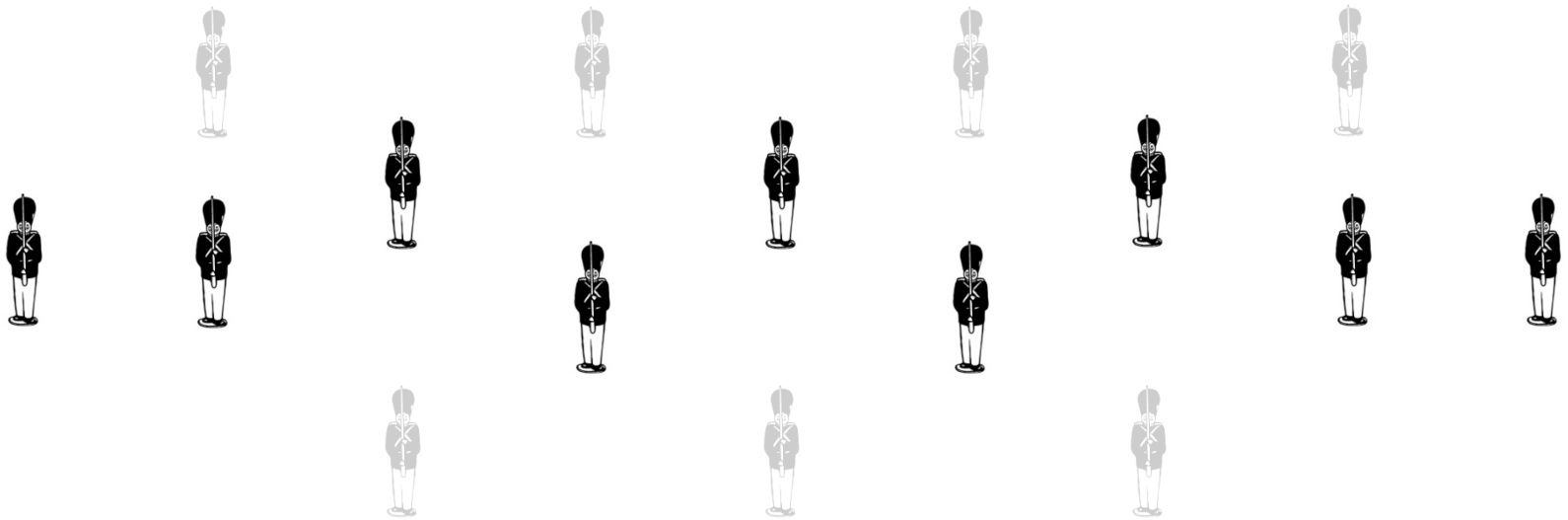




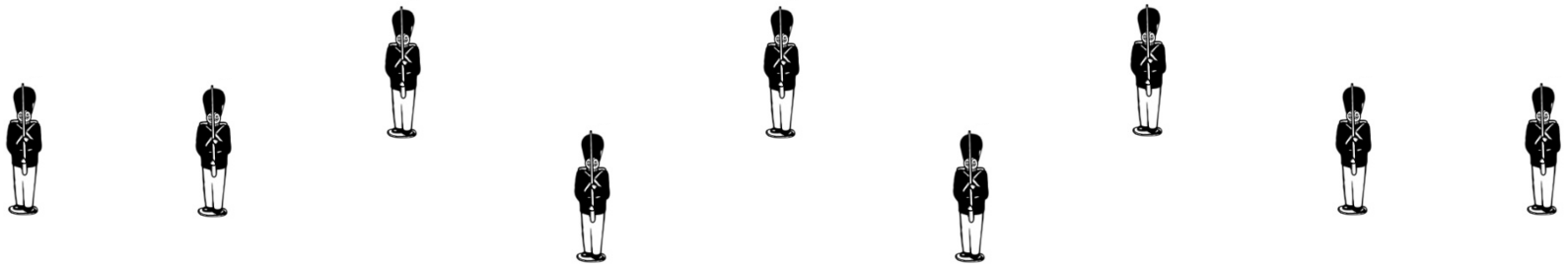


Slow convergence



Local solution: damping

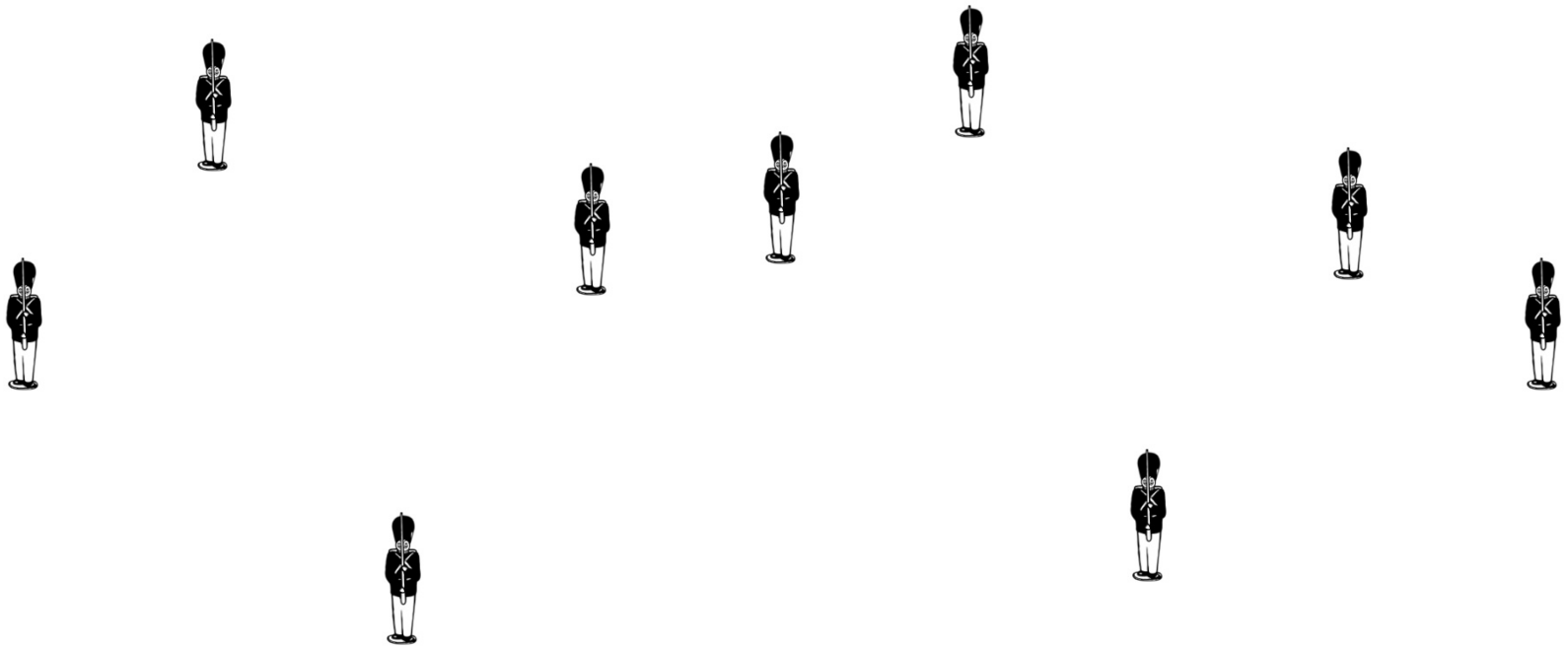


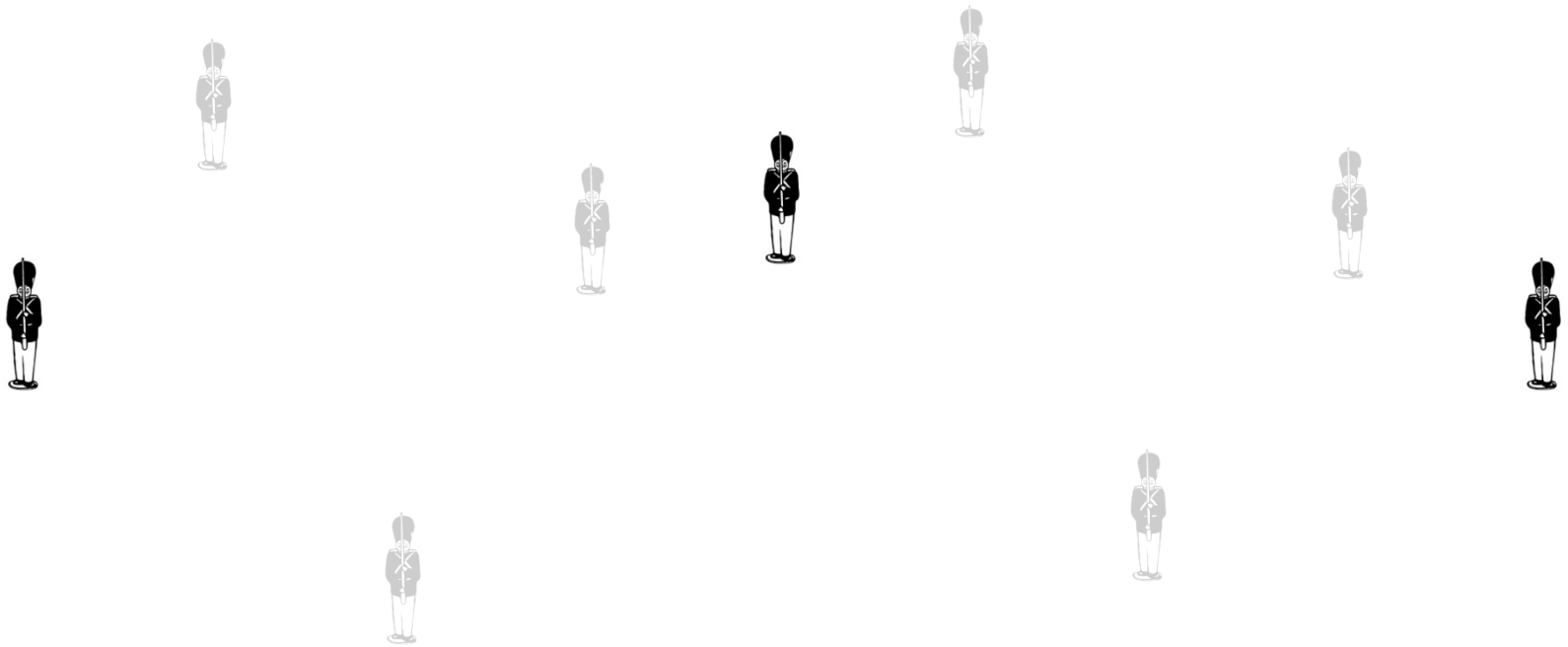
Local solution: damping

Local solution: damping

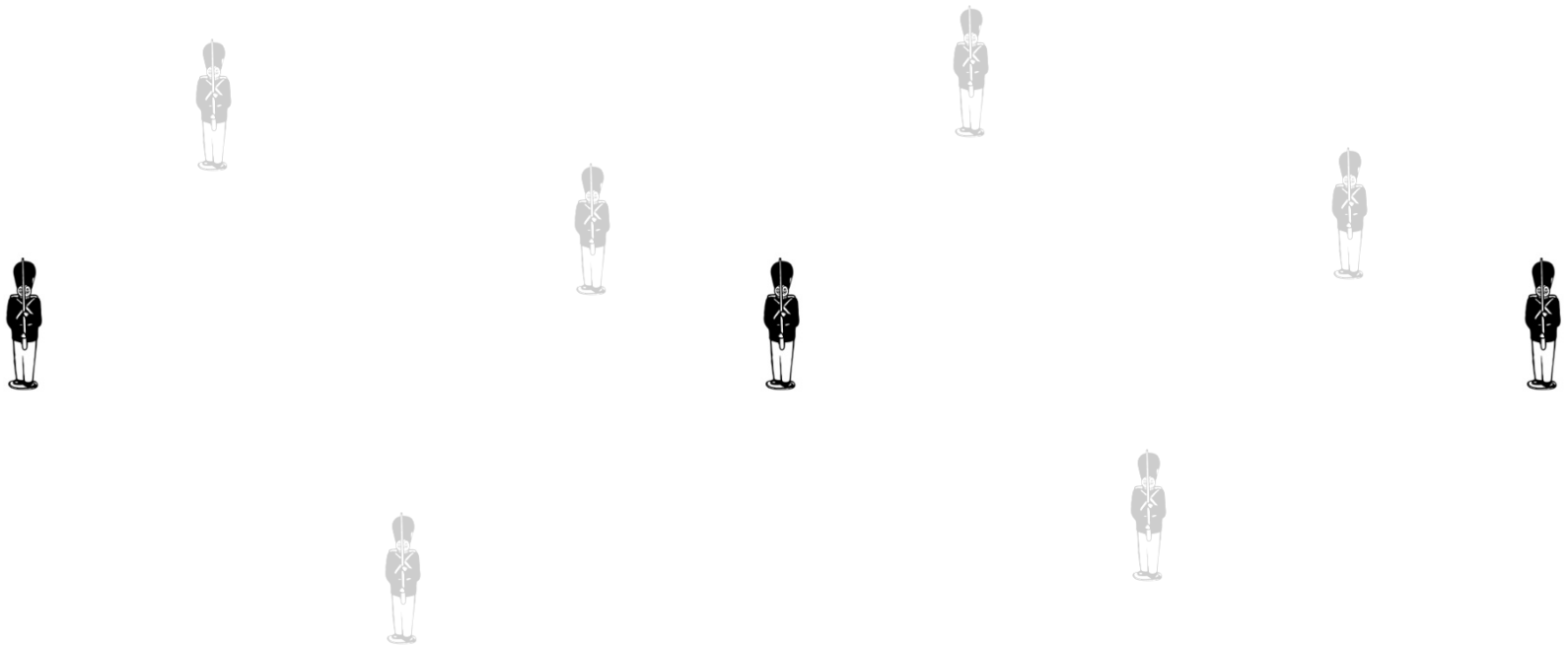
Local solution: damping

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

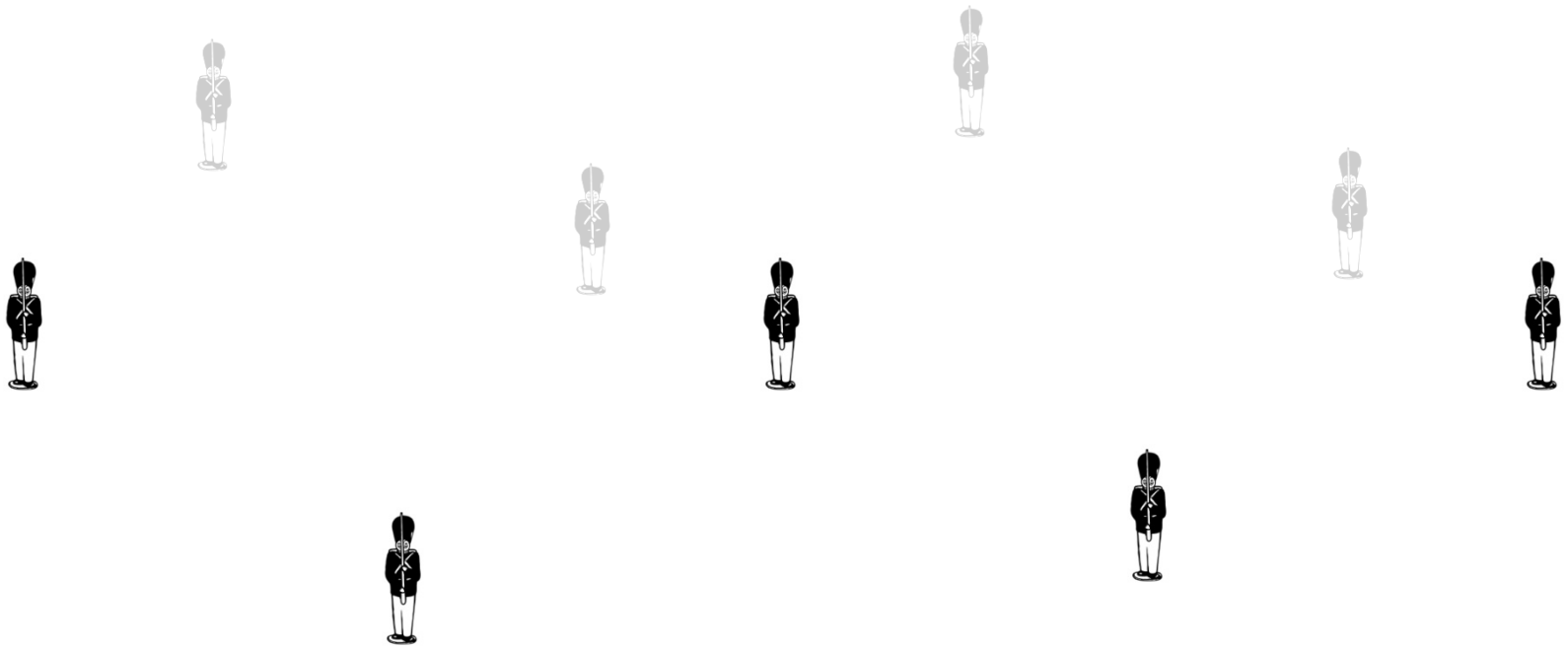




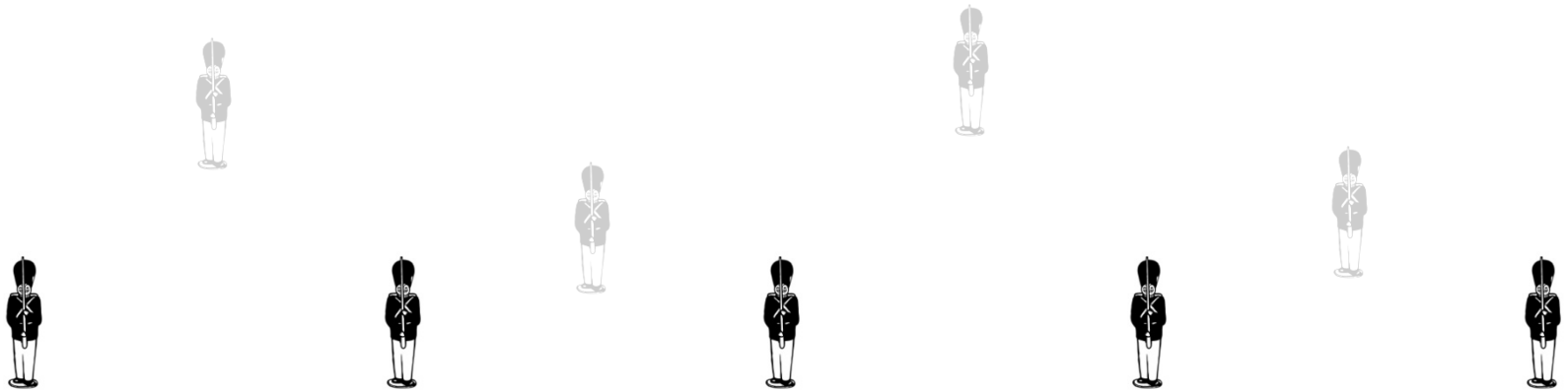
Large scale



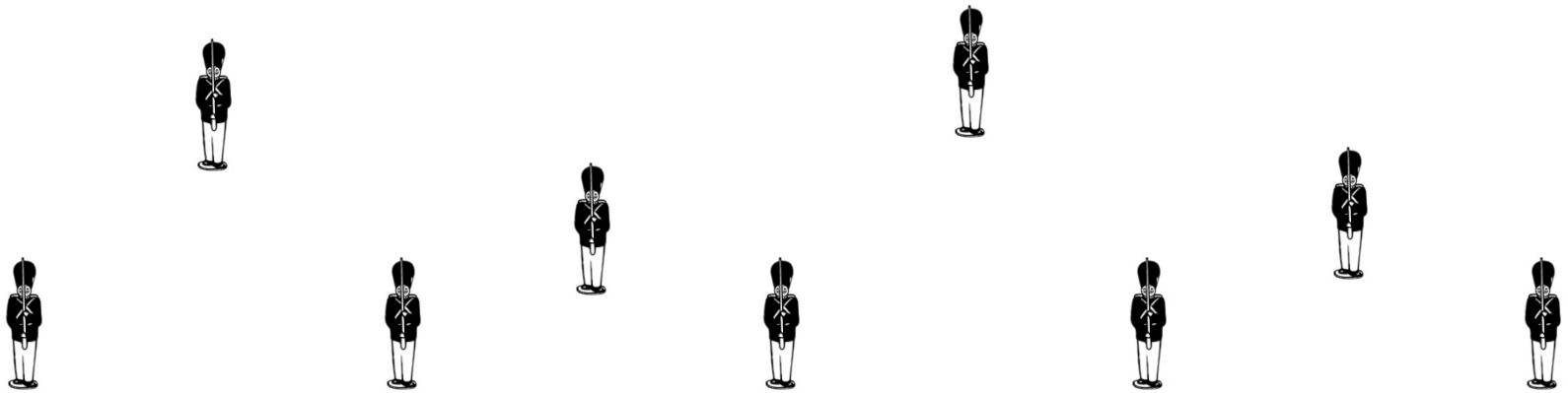
Large scale



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:

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

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

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

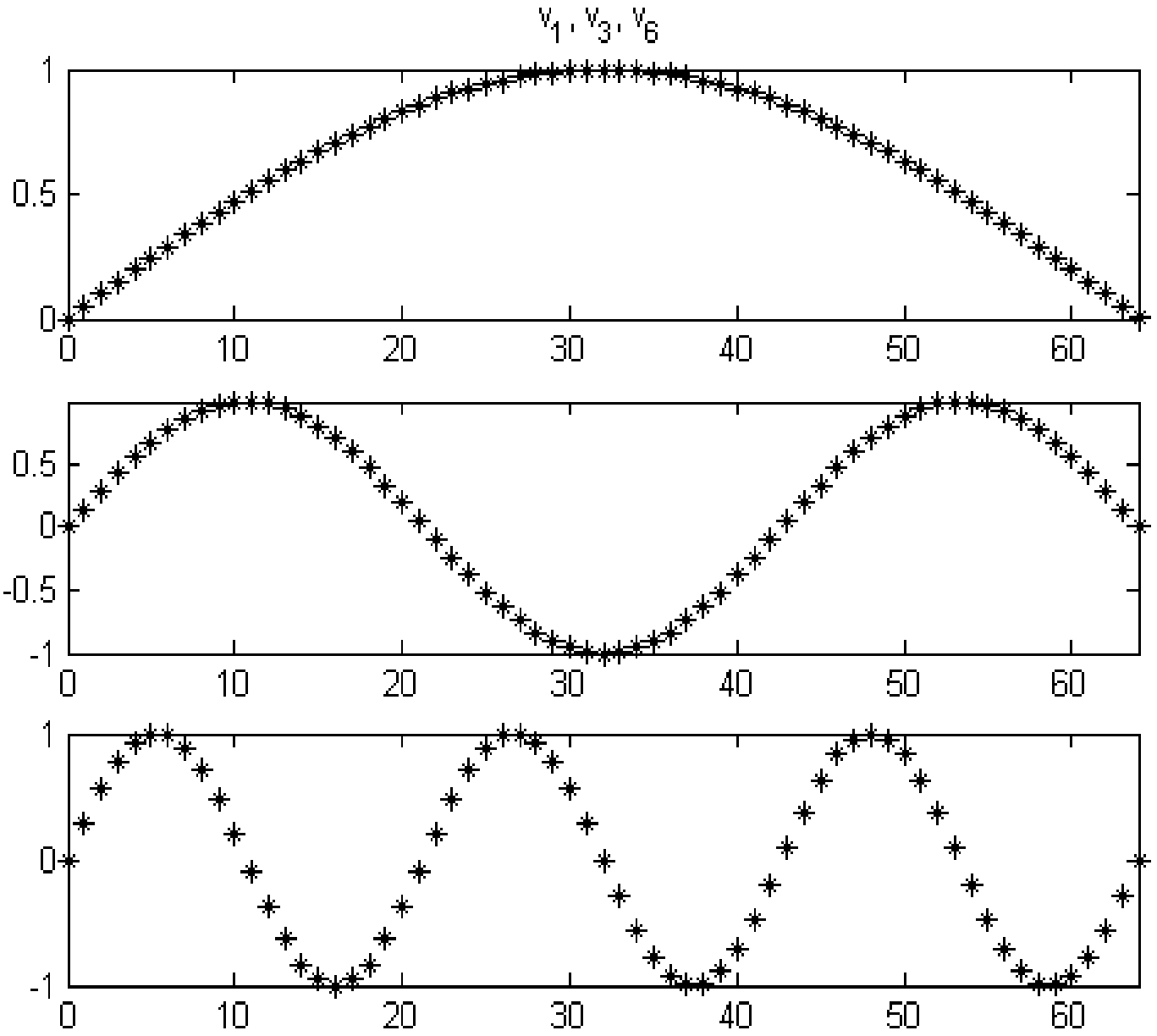
$$\lambda_k = \cos\left(\frac{k\pi}{N} \right),$$

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

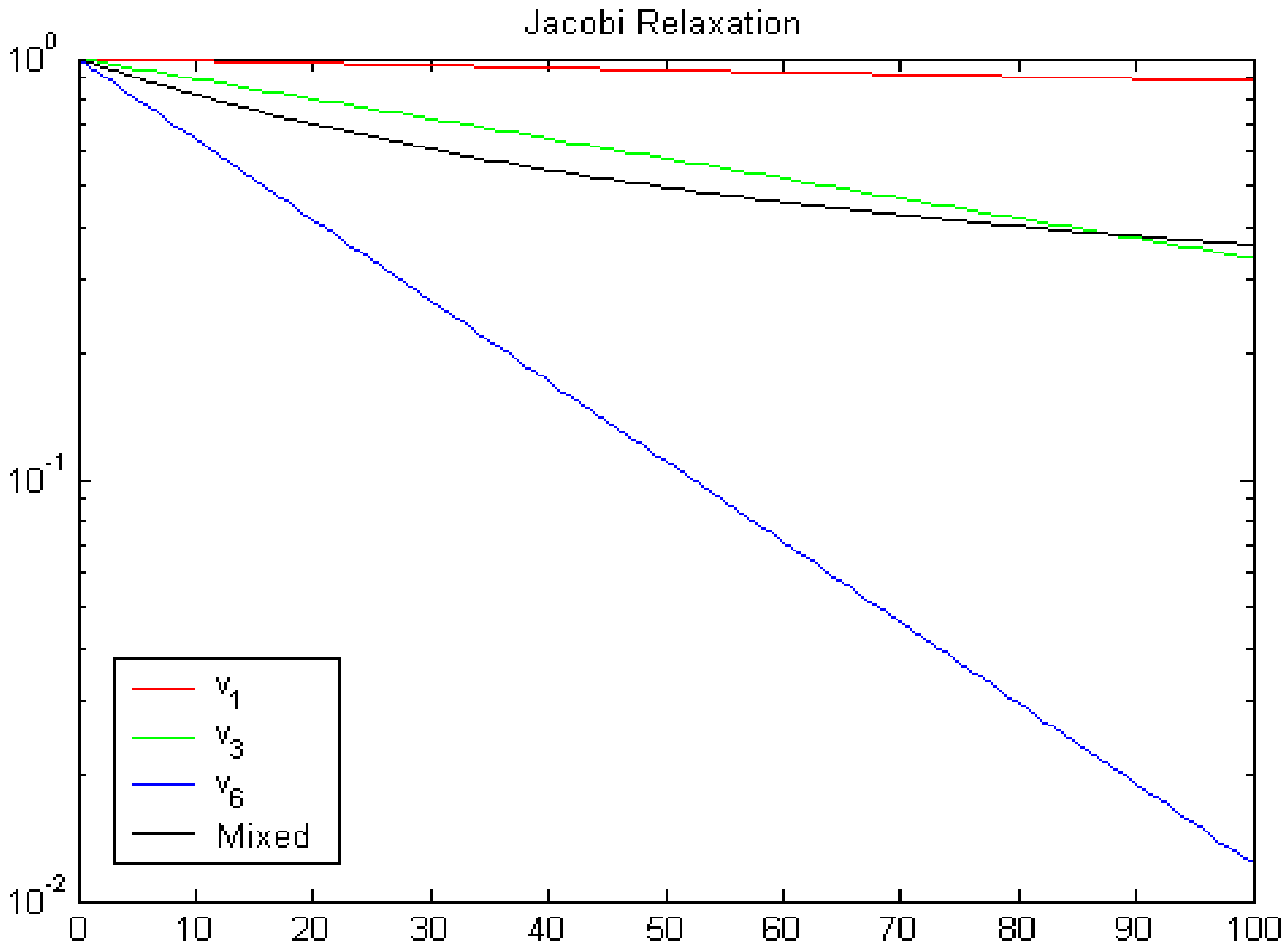
Note: since $|\lambda_k| < 1$, the method converges. But, for some eigenvectors, $|\lambda_k|$ is close to 1, so convergence is slow. In particular, for $k\pi/N << 1$, we have,

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

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

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 $k / N \approx 1.$

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

Definition:

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

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

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

$$\lambda_k \le 0.$$

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?

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:

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

Derivation: by the Taylor theorem

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

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

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

$$u_0^h = u_0,$$

$$u_N^h = u_1.$$

In matrix form:

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

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

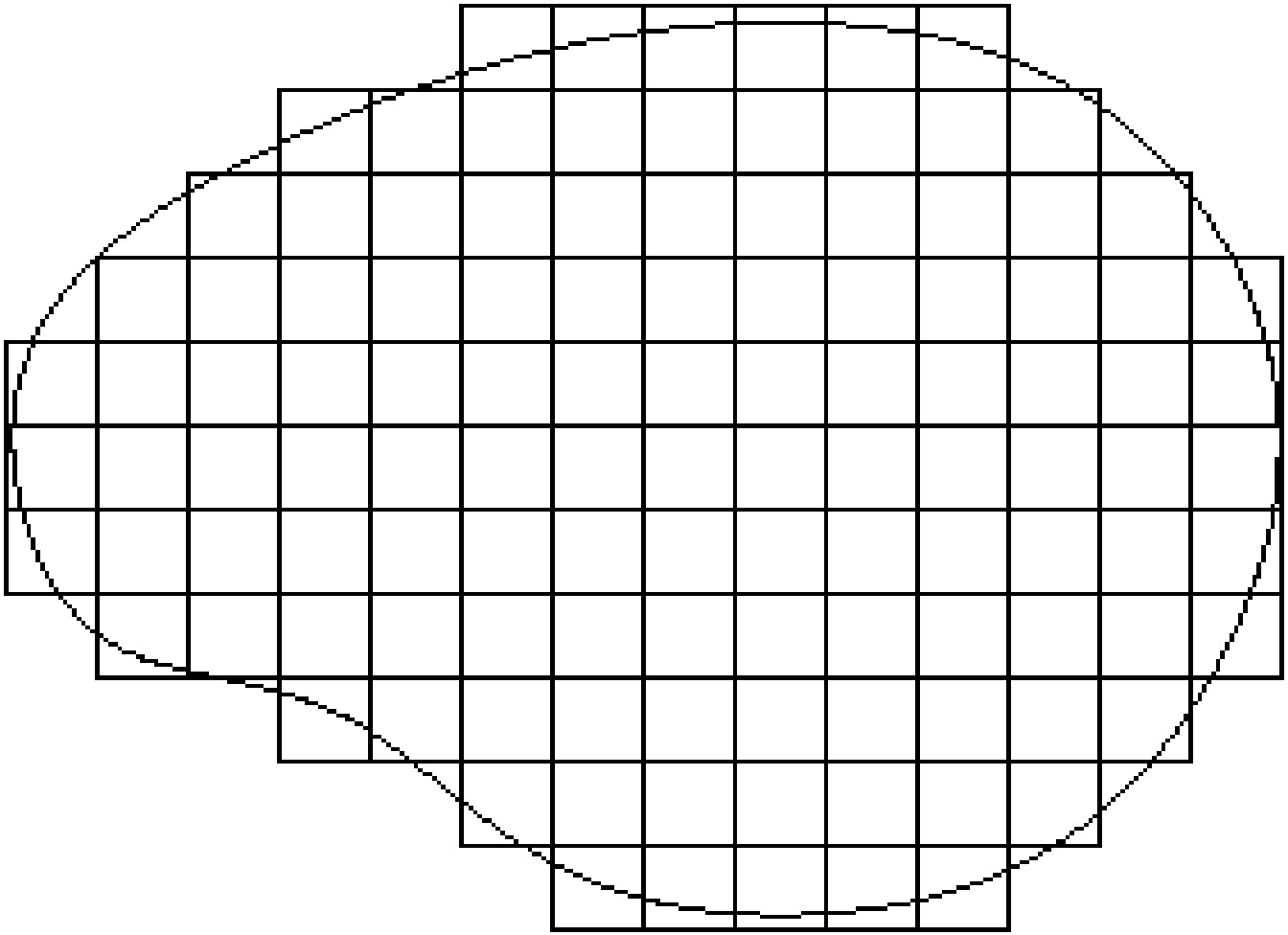
This is a tridiagonal system of equations that is essentially as easy to solve as the soldier problem

2D Model Problem

Find u which satisfies:

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

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:

$$L^h u^h_{i,j} = \qquad (5)$$

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

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

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

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

Gauss-Seidel (GS) Relaxation:

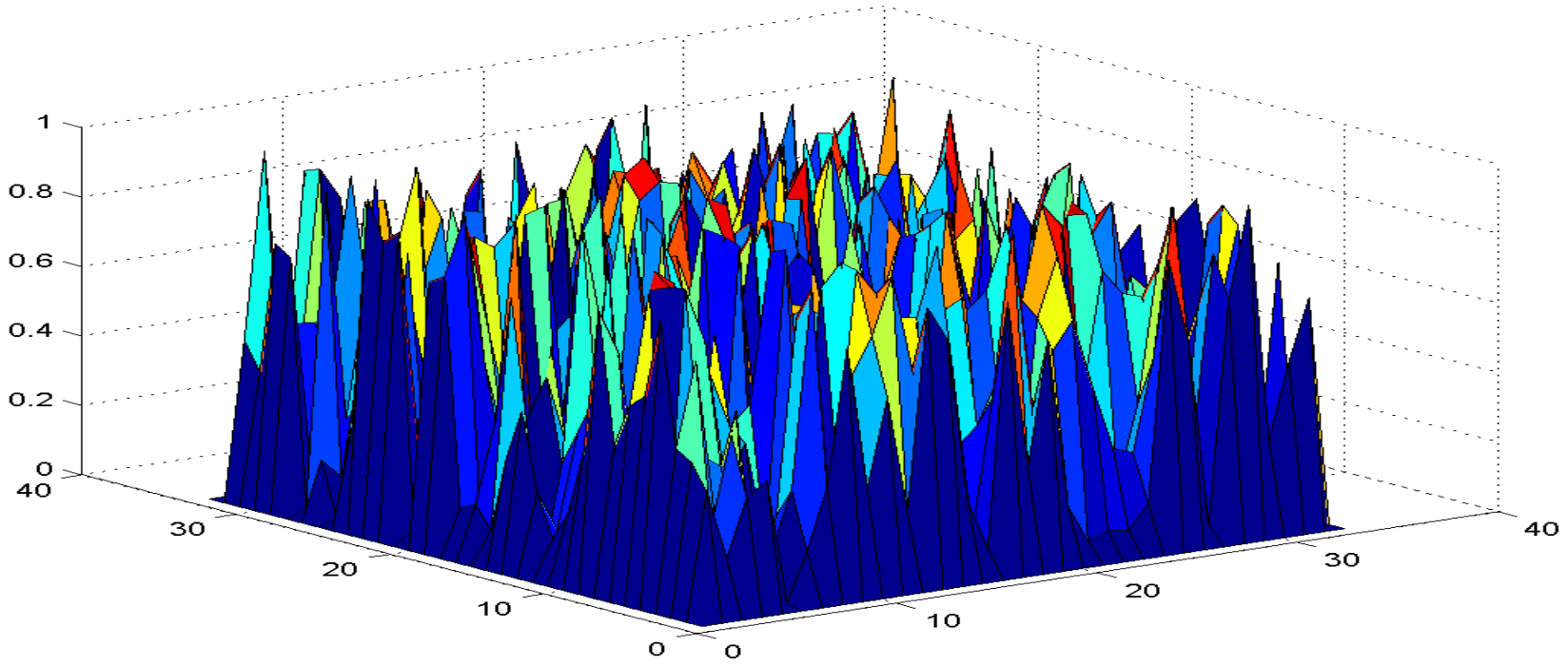
1. Choose initial guess, $\tilde{u}^h$.
2. Repeat until some convergence criterion is satisfied
   {

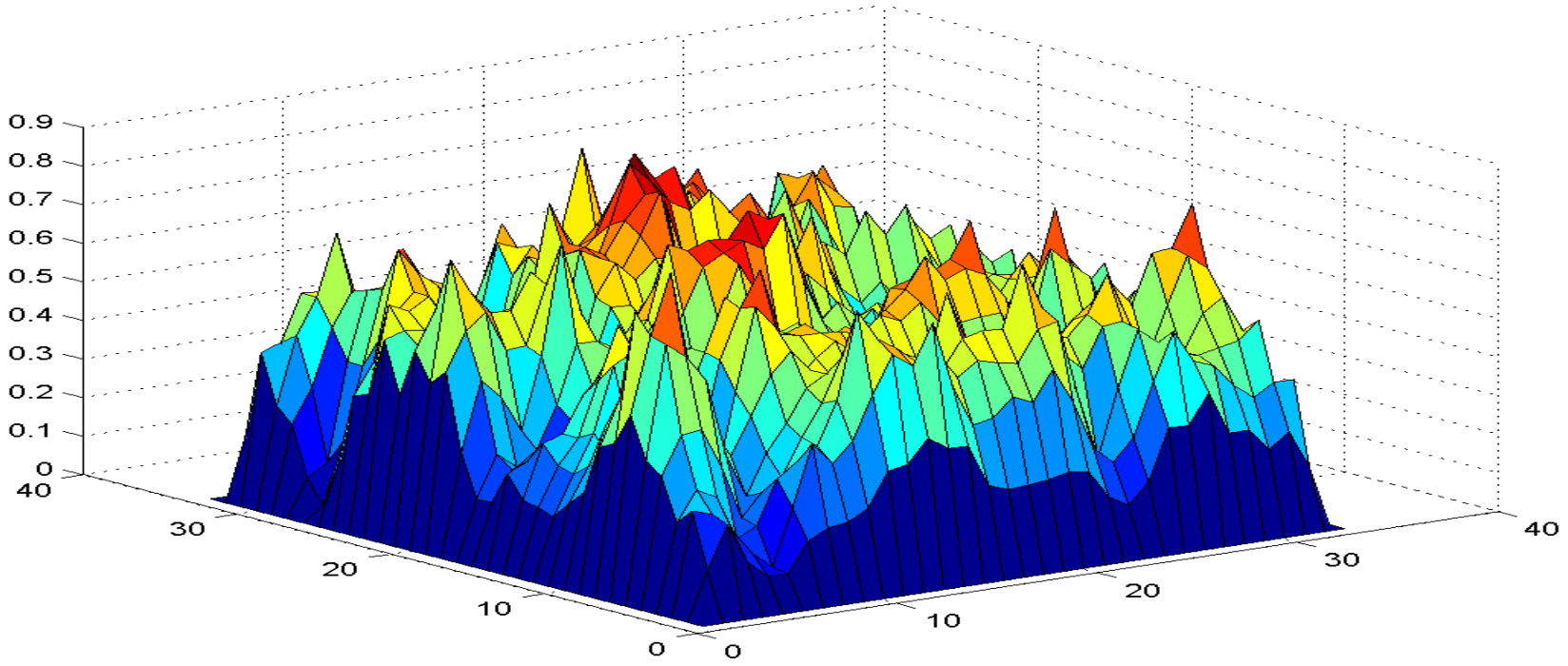
   Scan all variables in some prescribed order, and change each variable $\tilde{u}^h_{i,j}$ in turn so as to satisfy the (i,j)th equation.

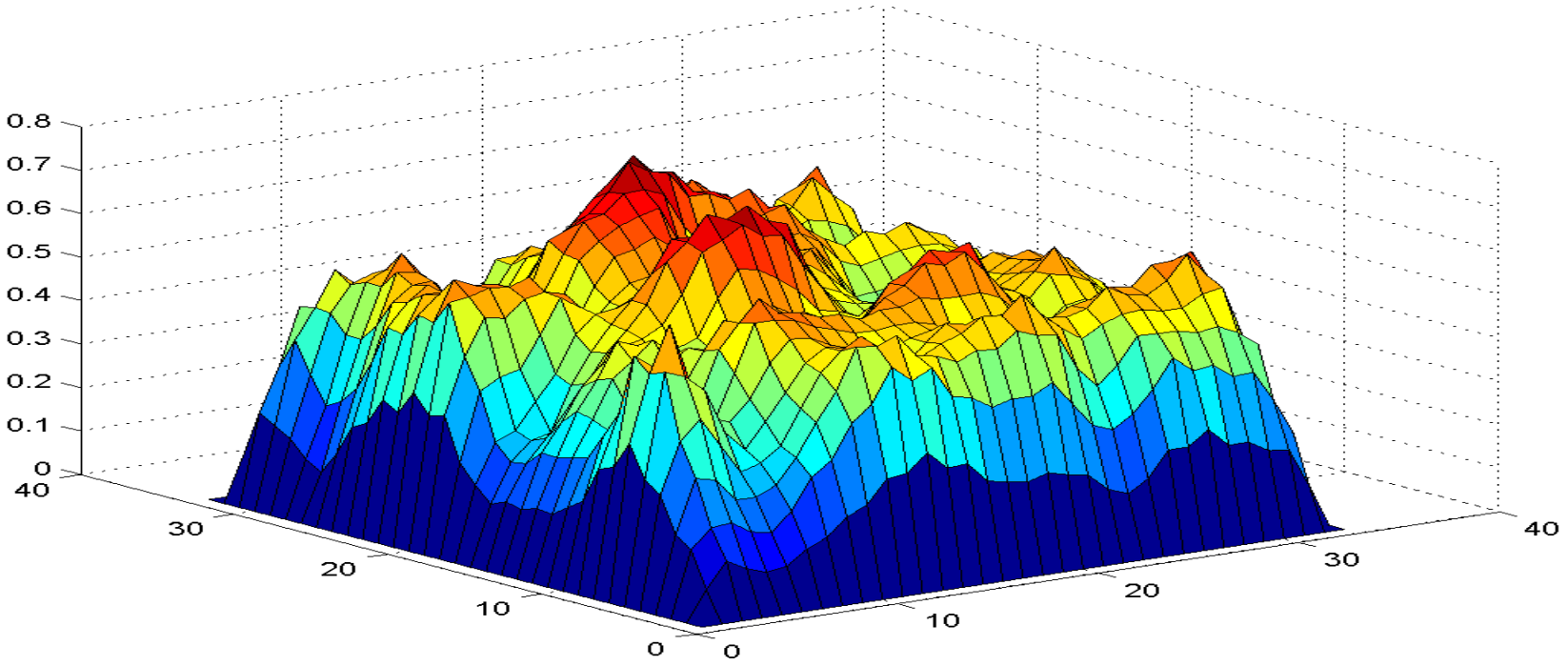
   }

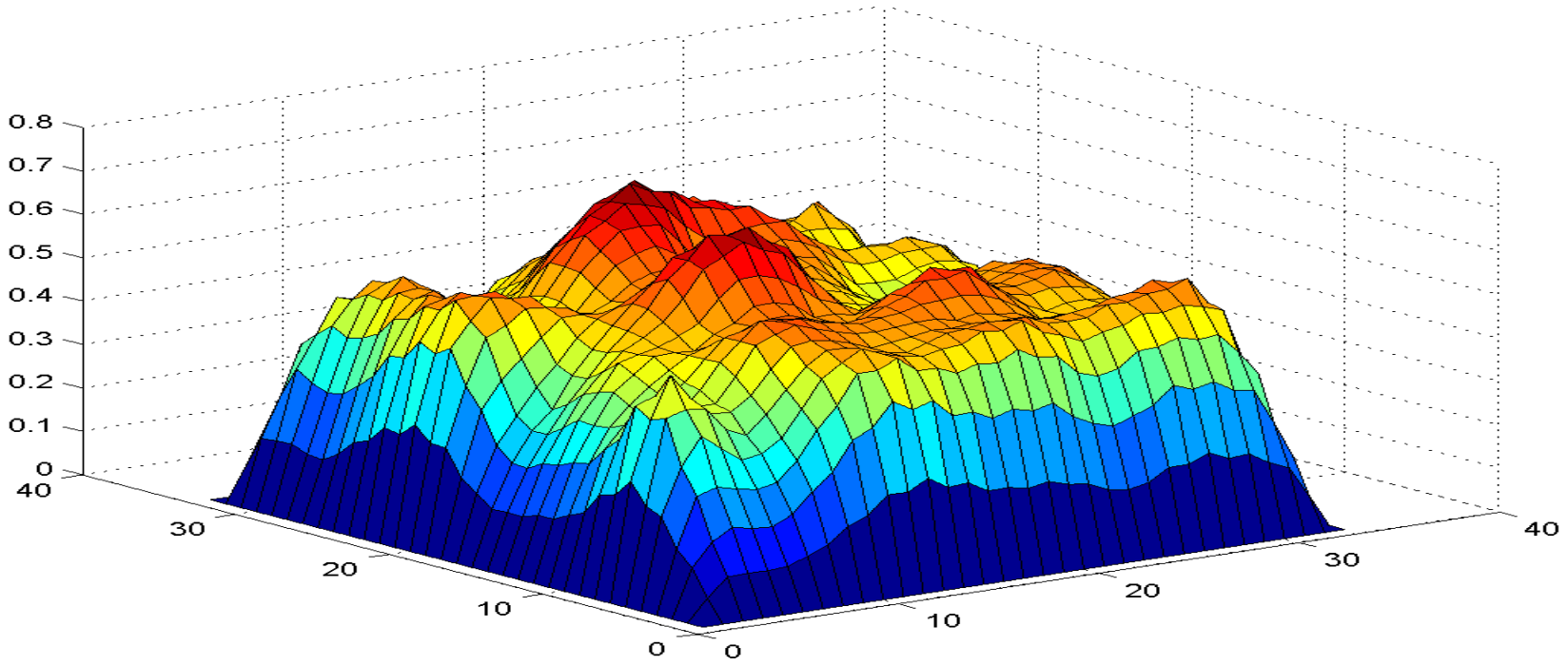
Observation: GS is a local process, because only near neighbors appear in each equation. Hence, it may be efficient for eliminating errors which can be detected locally. But large-scale ("smooth") errors are eliminated very slowly.

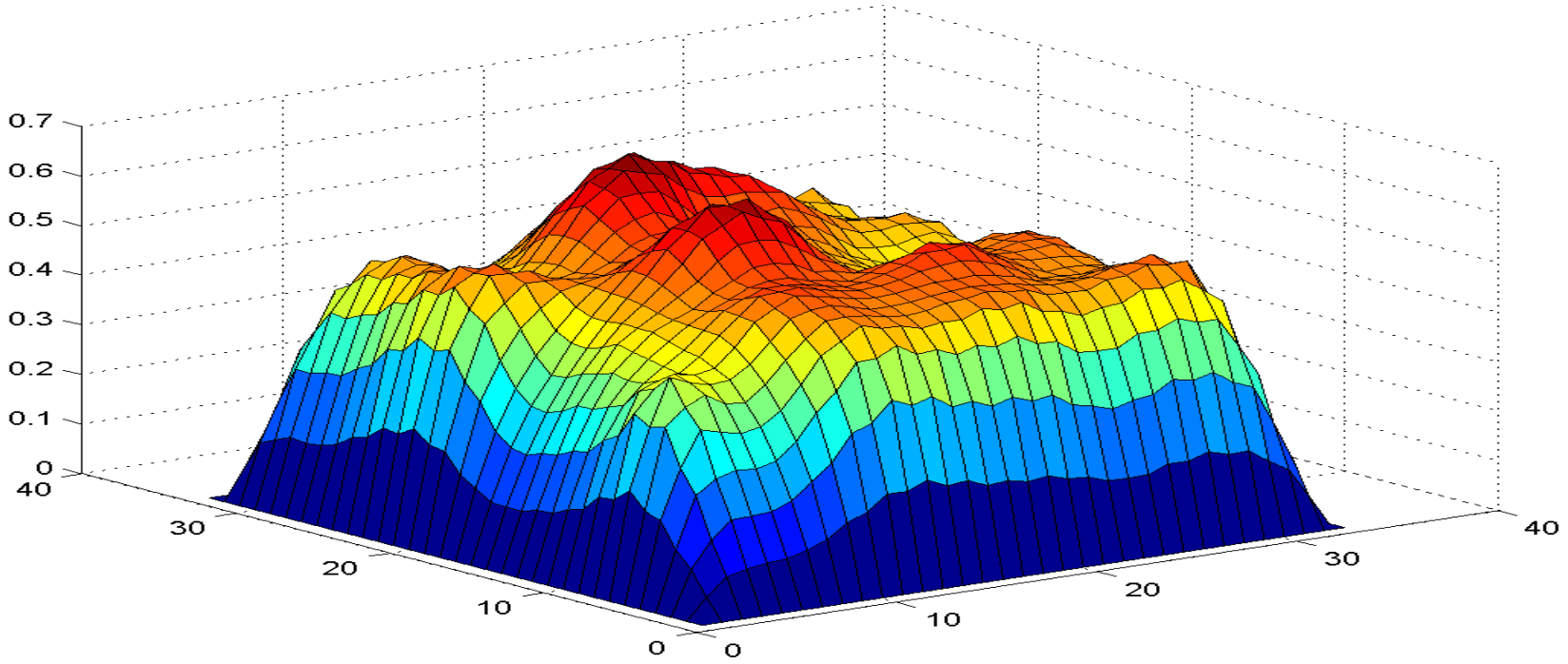
(The difference between GS and Jacobi is that old neighboring values are used in Jacobi, while the most updated values are used in GS.)

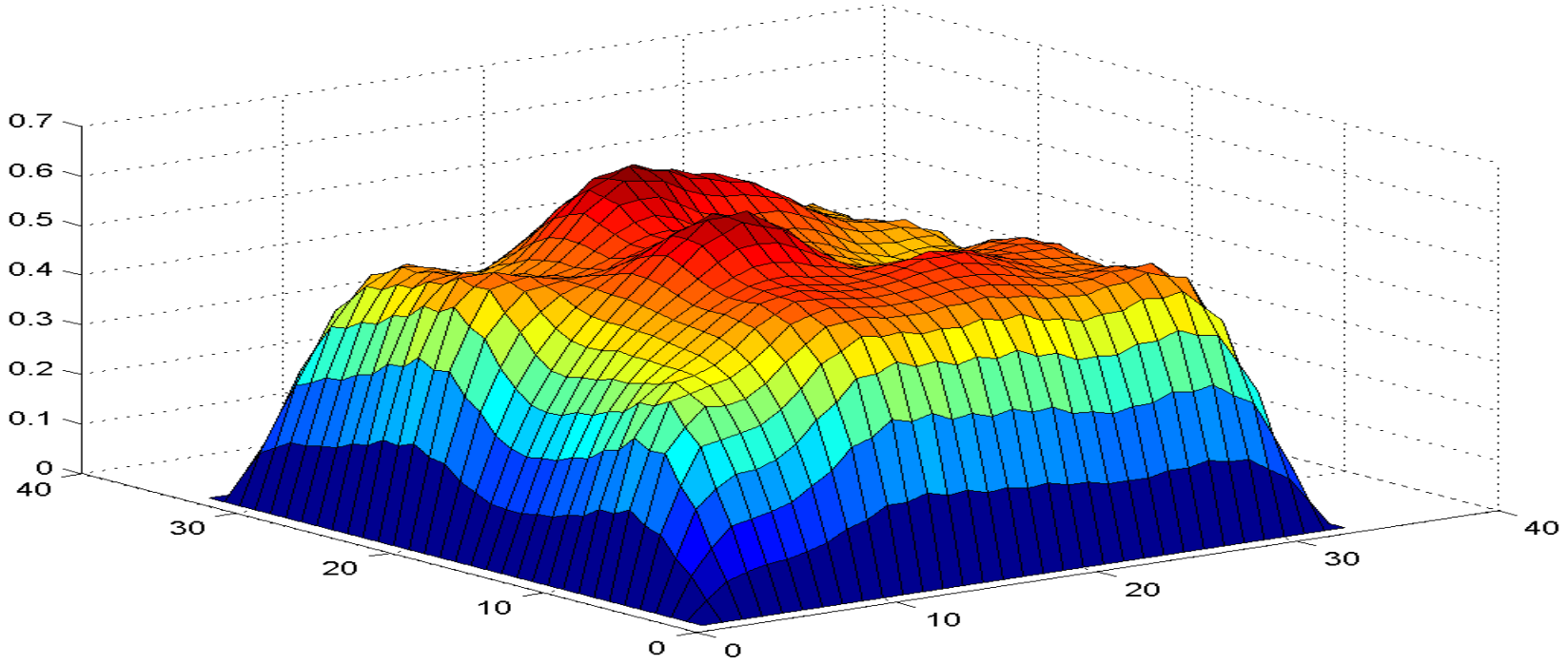

1
0.8
0.6
0.4
0.2
0
40
30
20
10
0
0
10
20
30
40

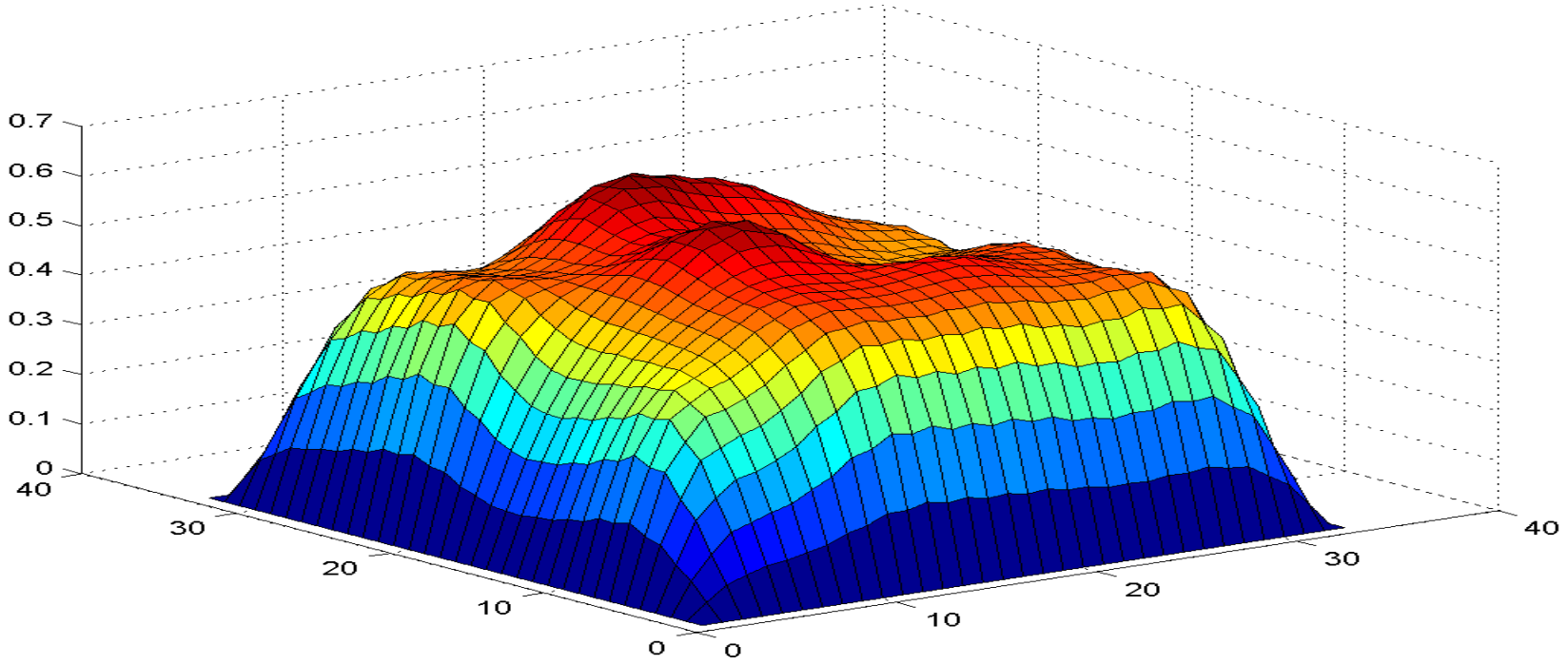

0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

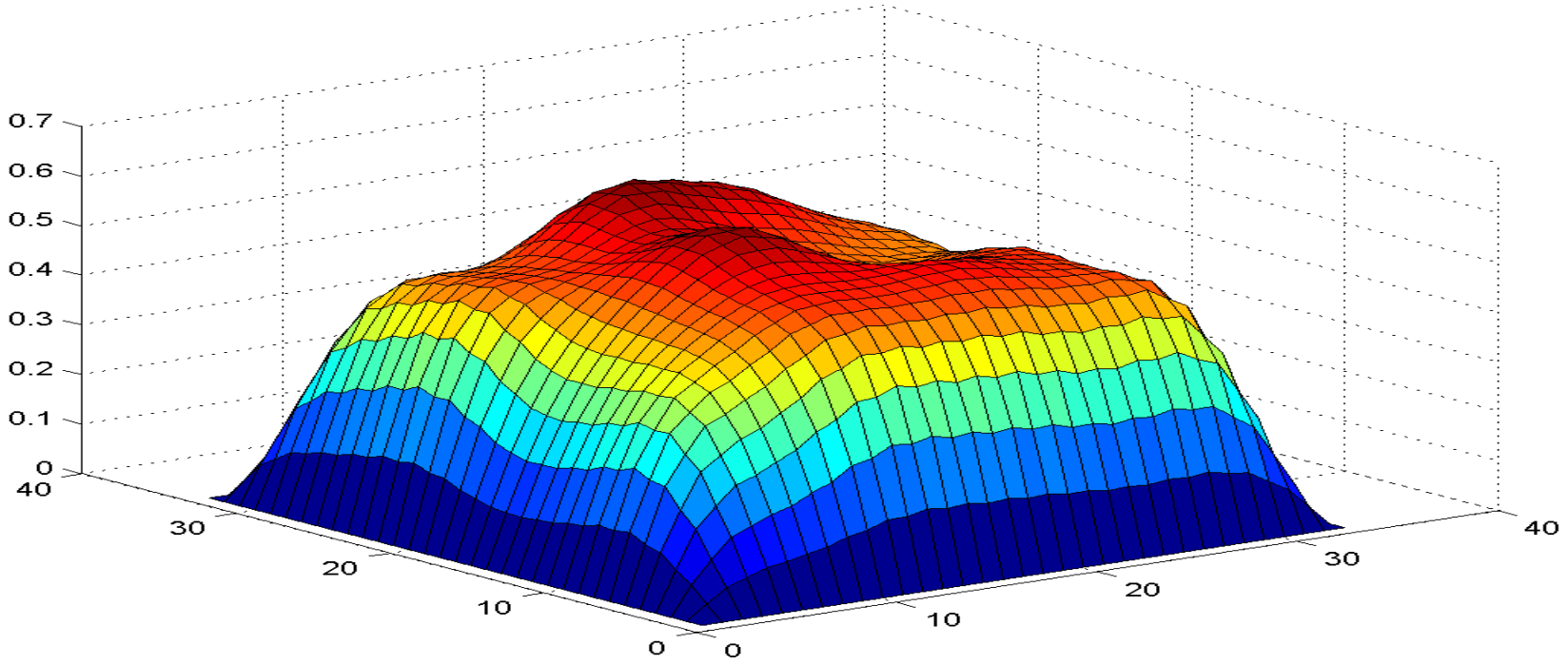

0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

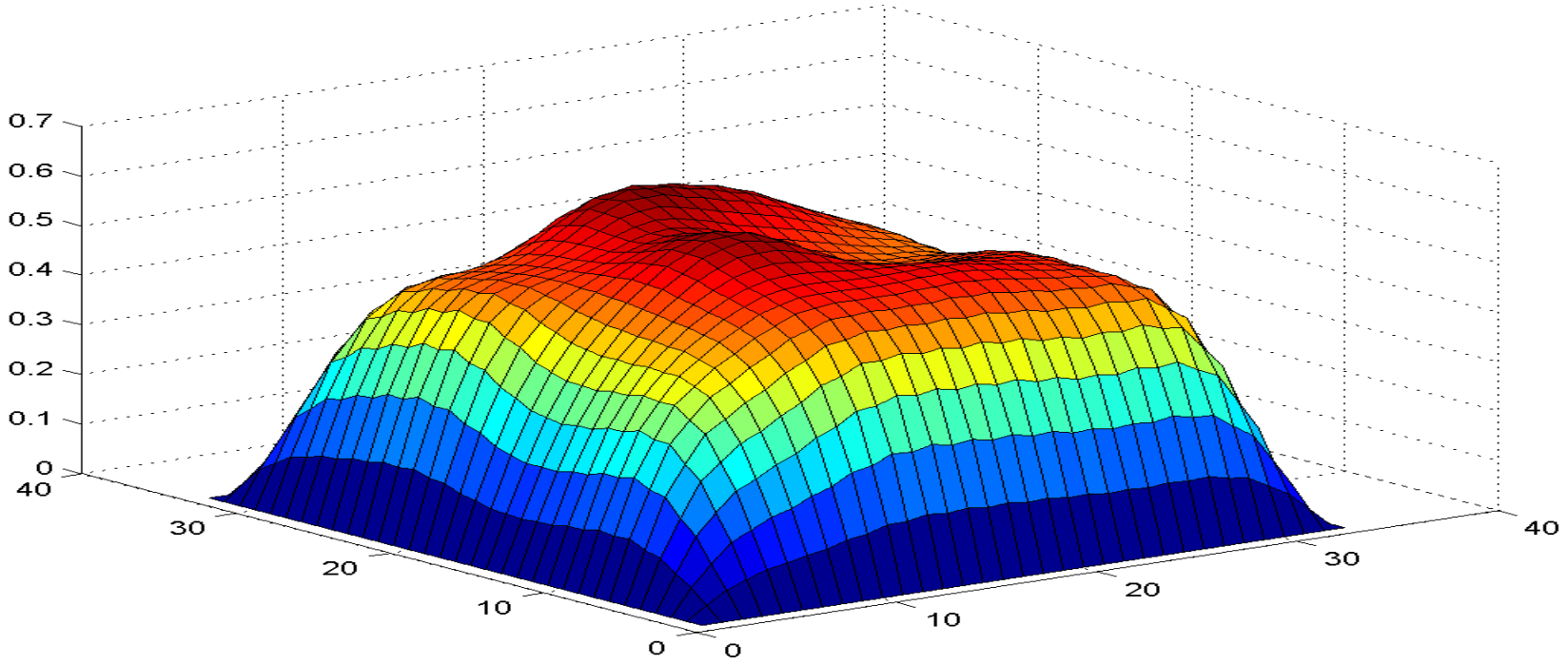

0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

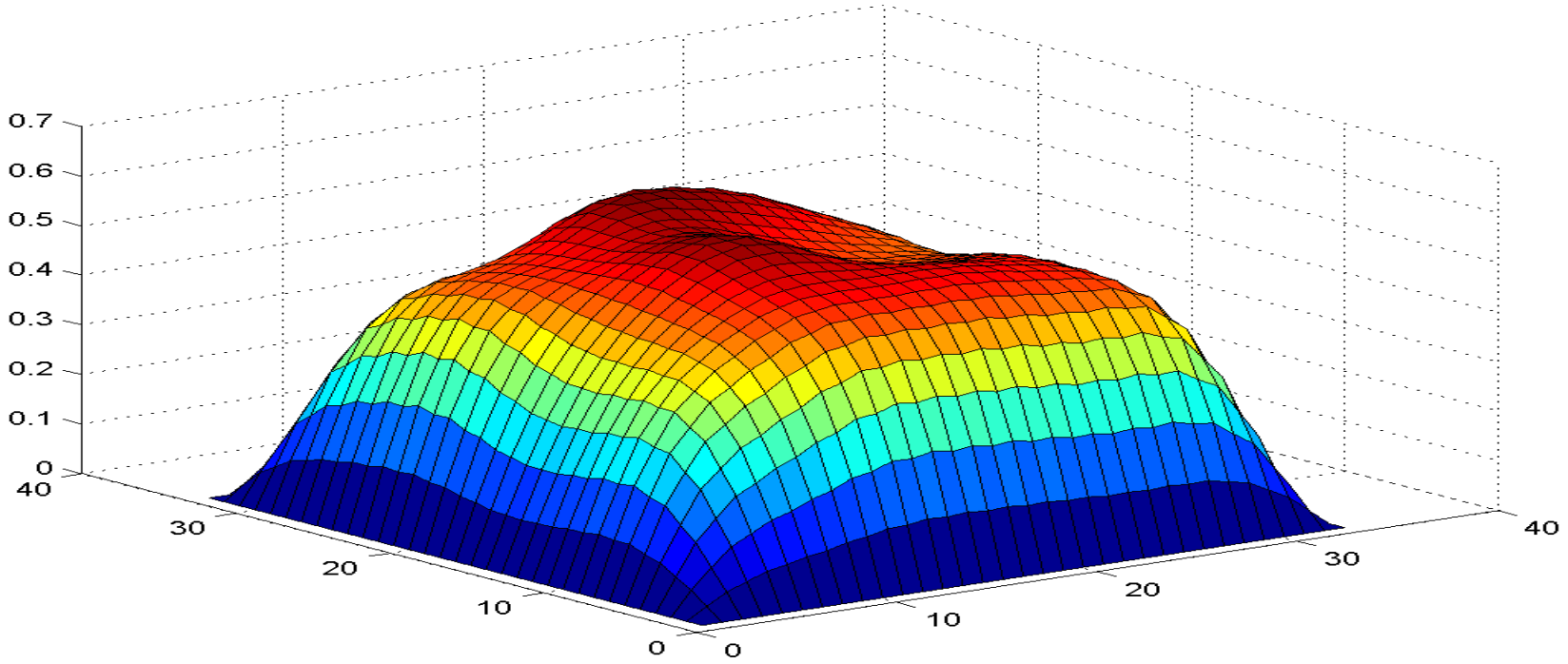

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

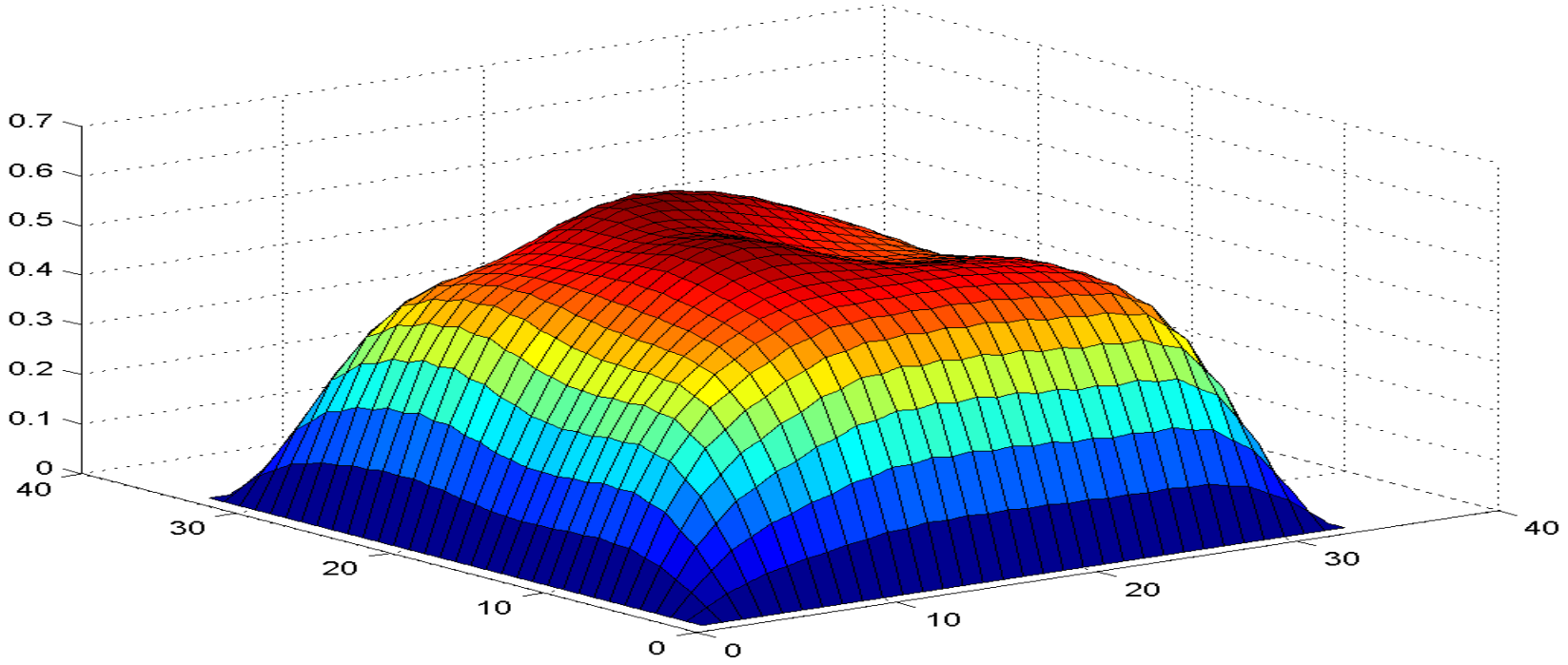

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

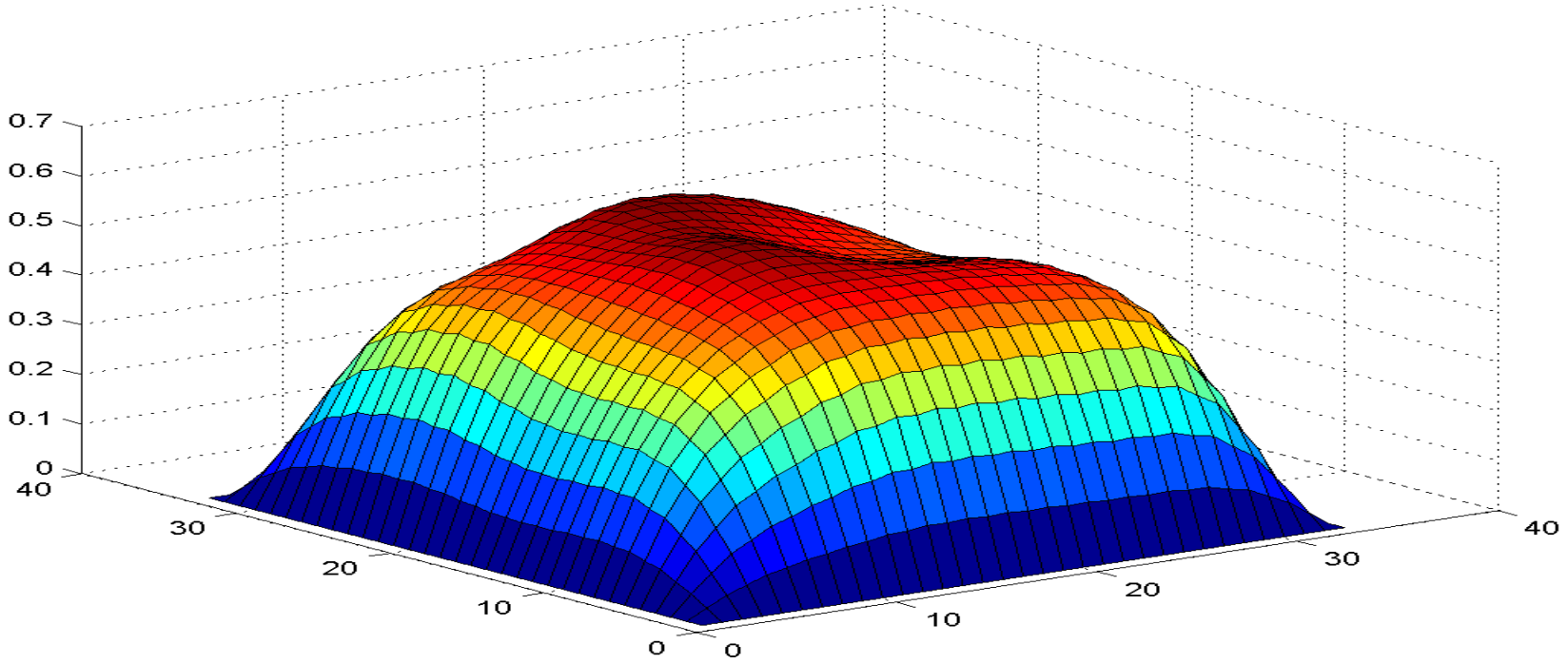

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

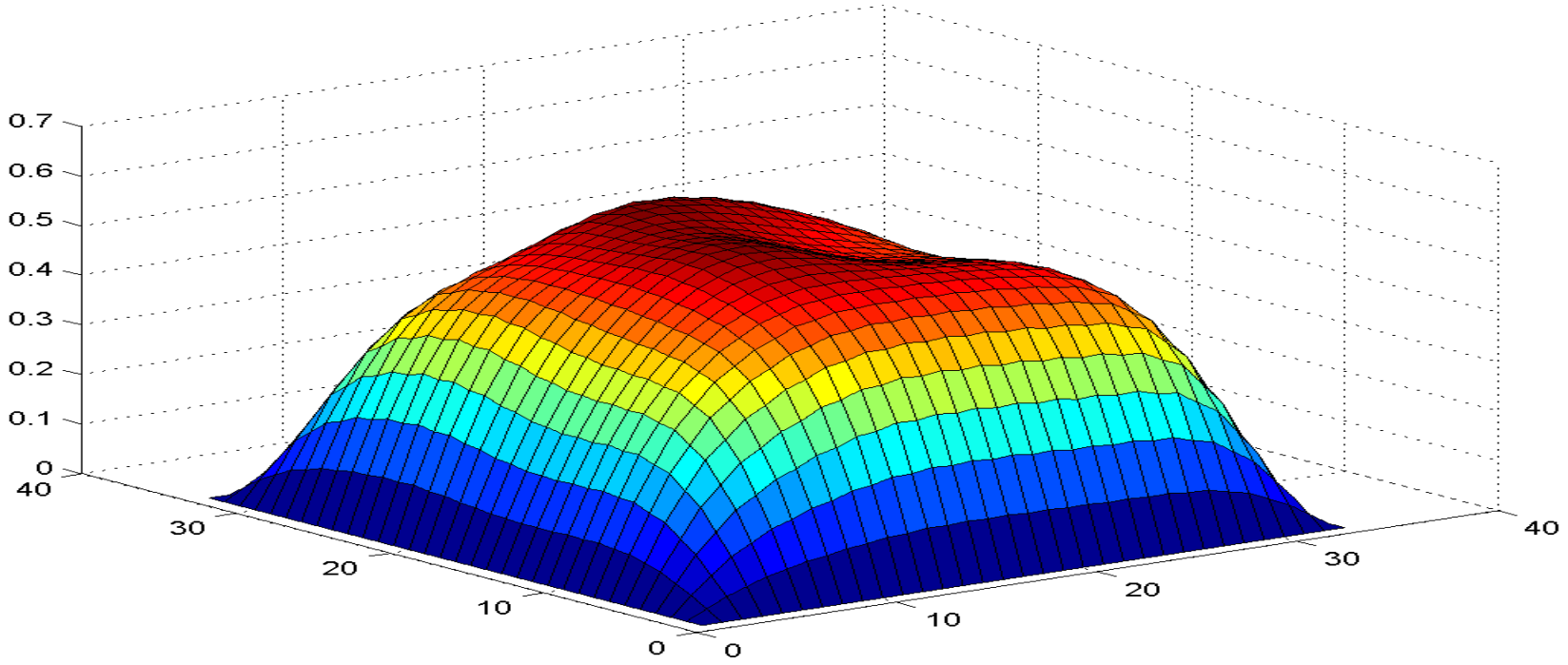

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

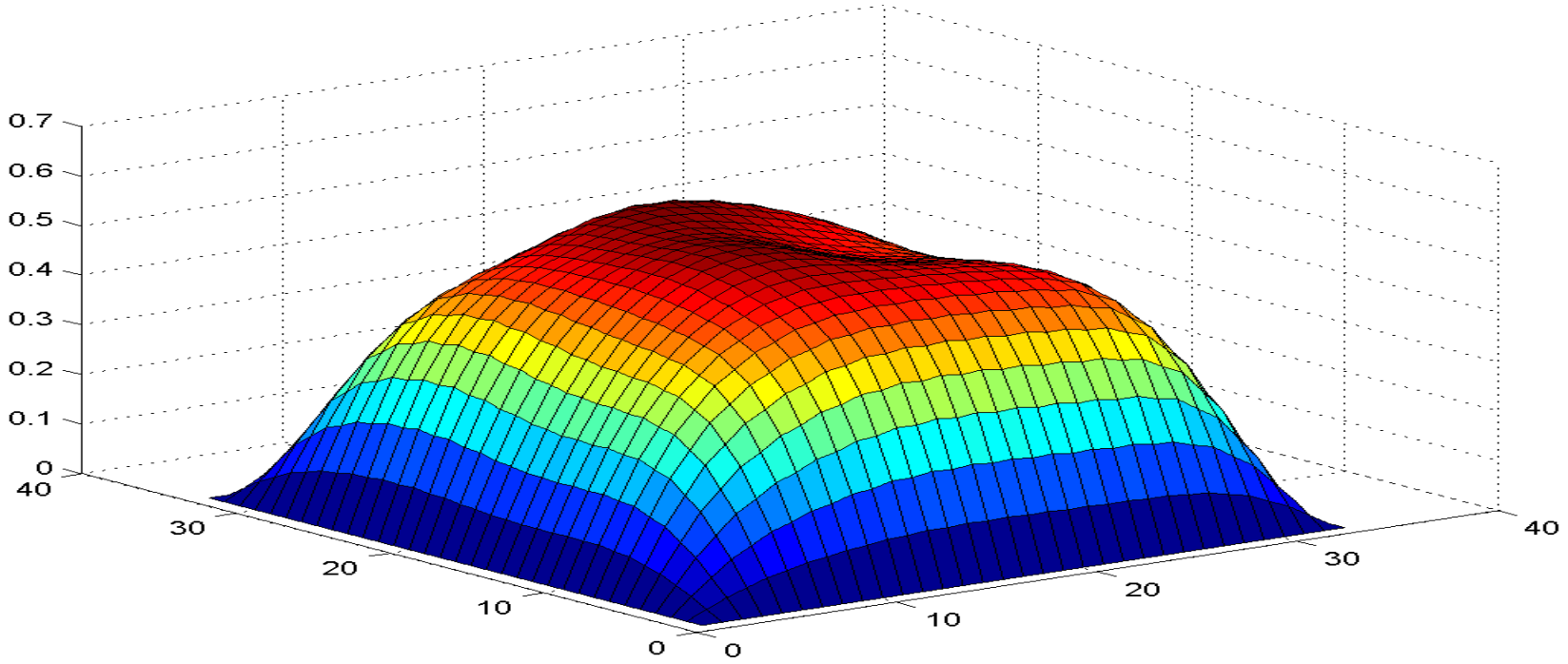

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

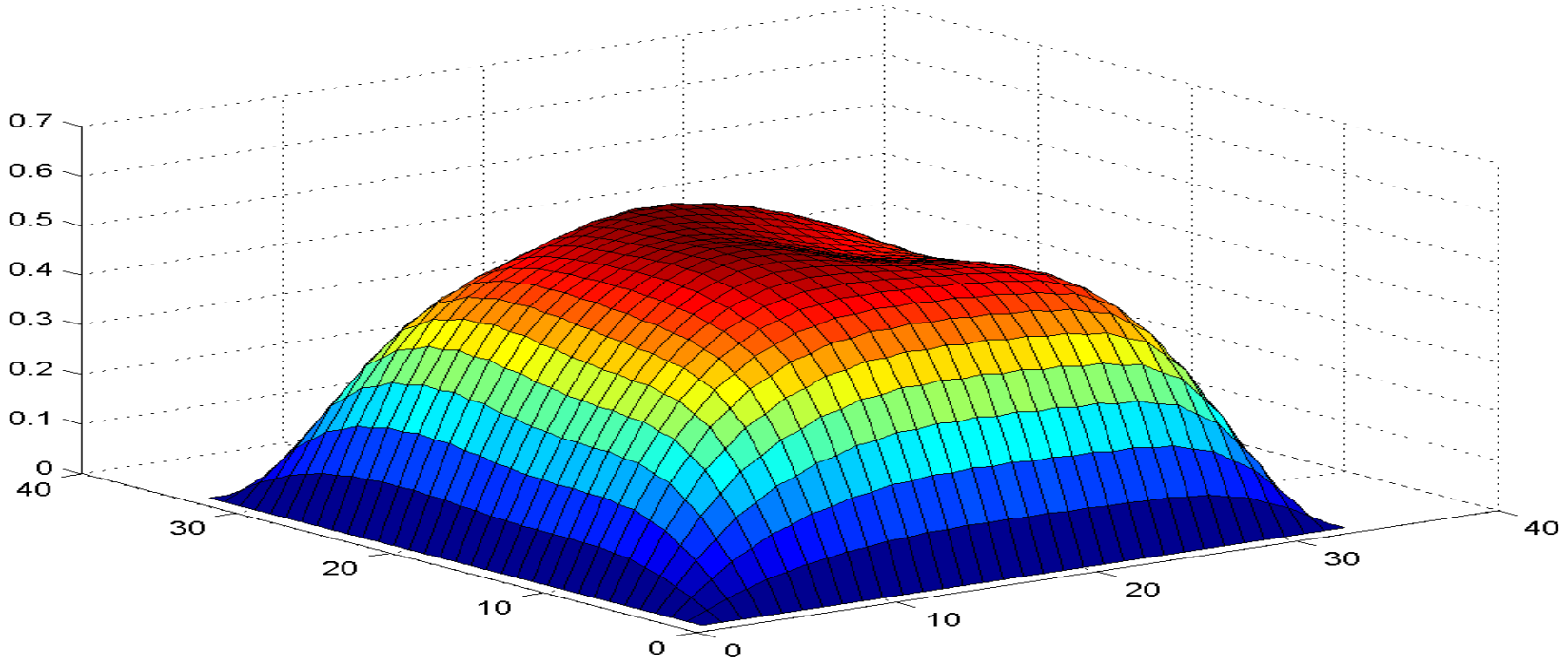

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

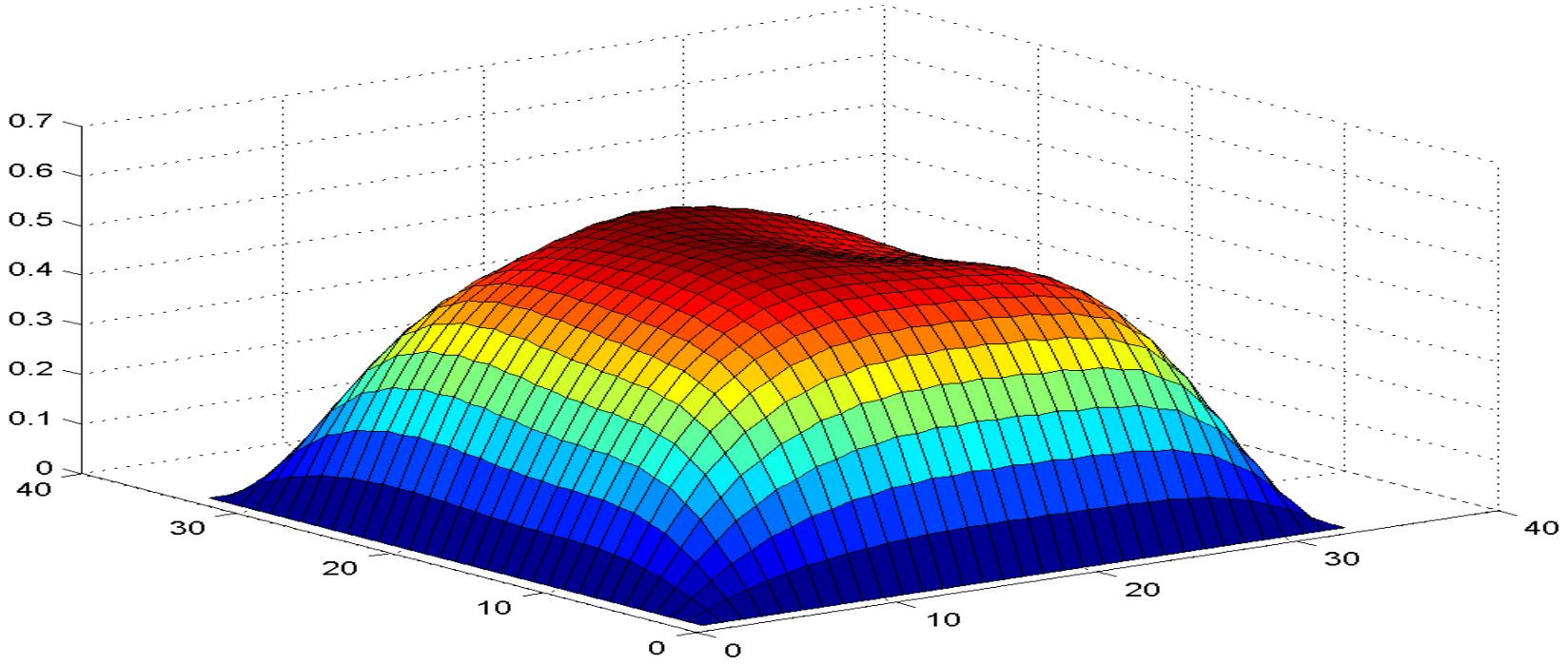

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

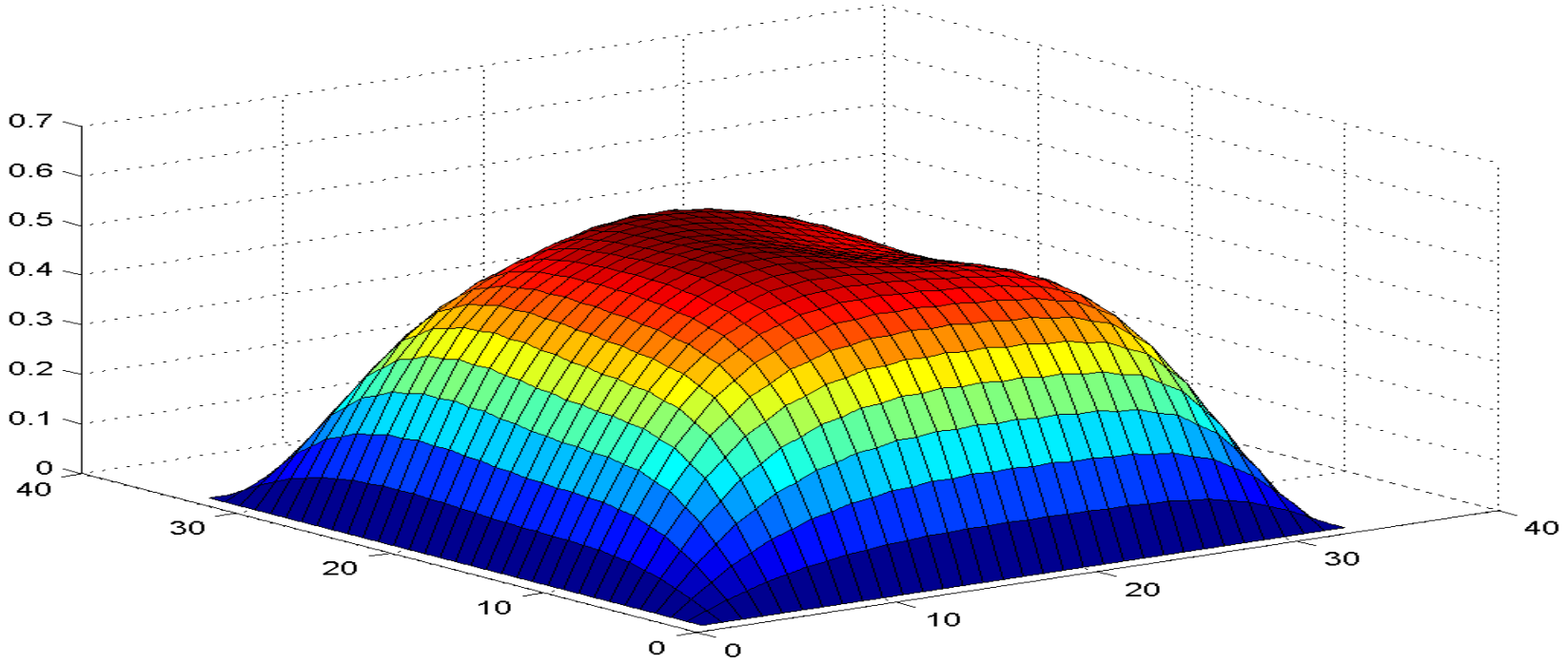

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

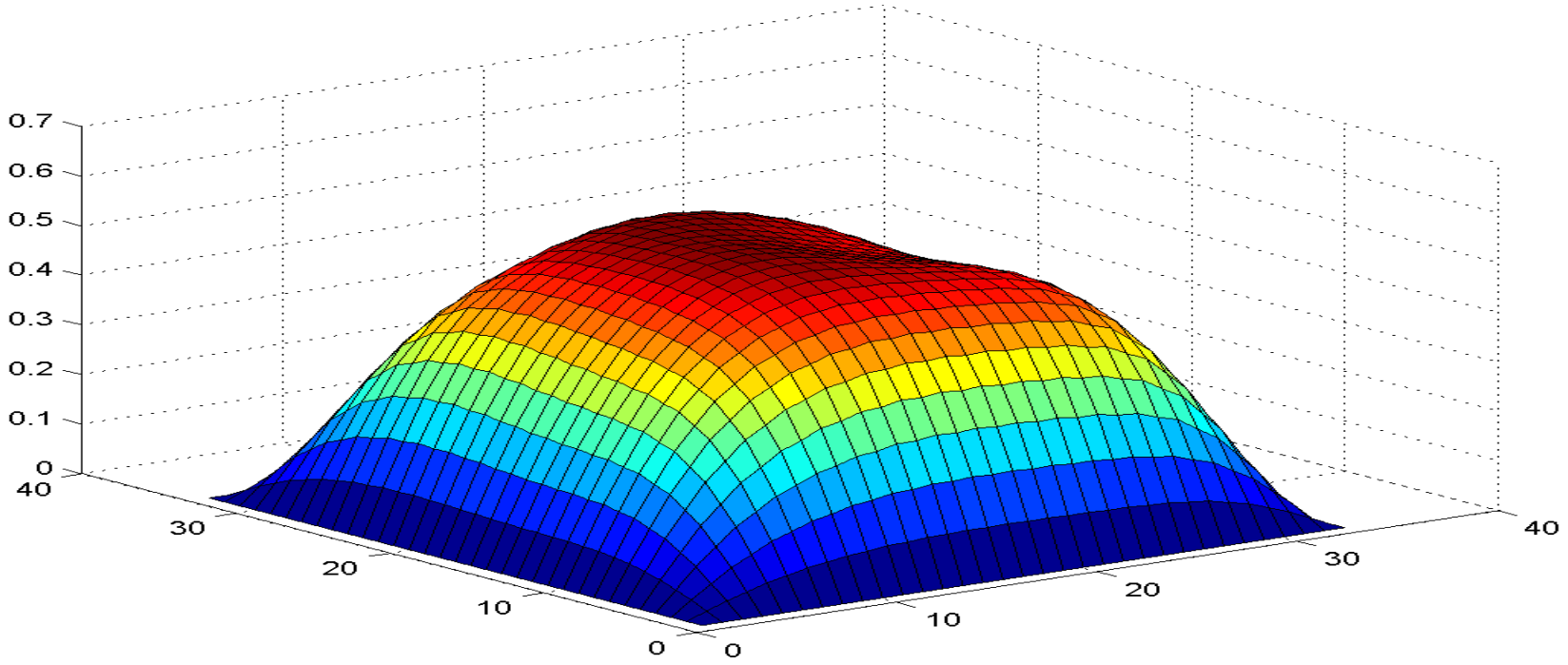

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

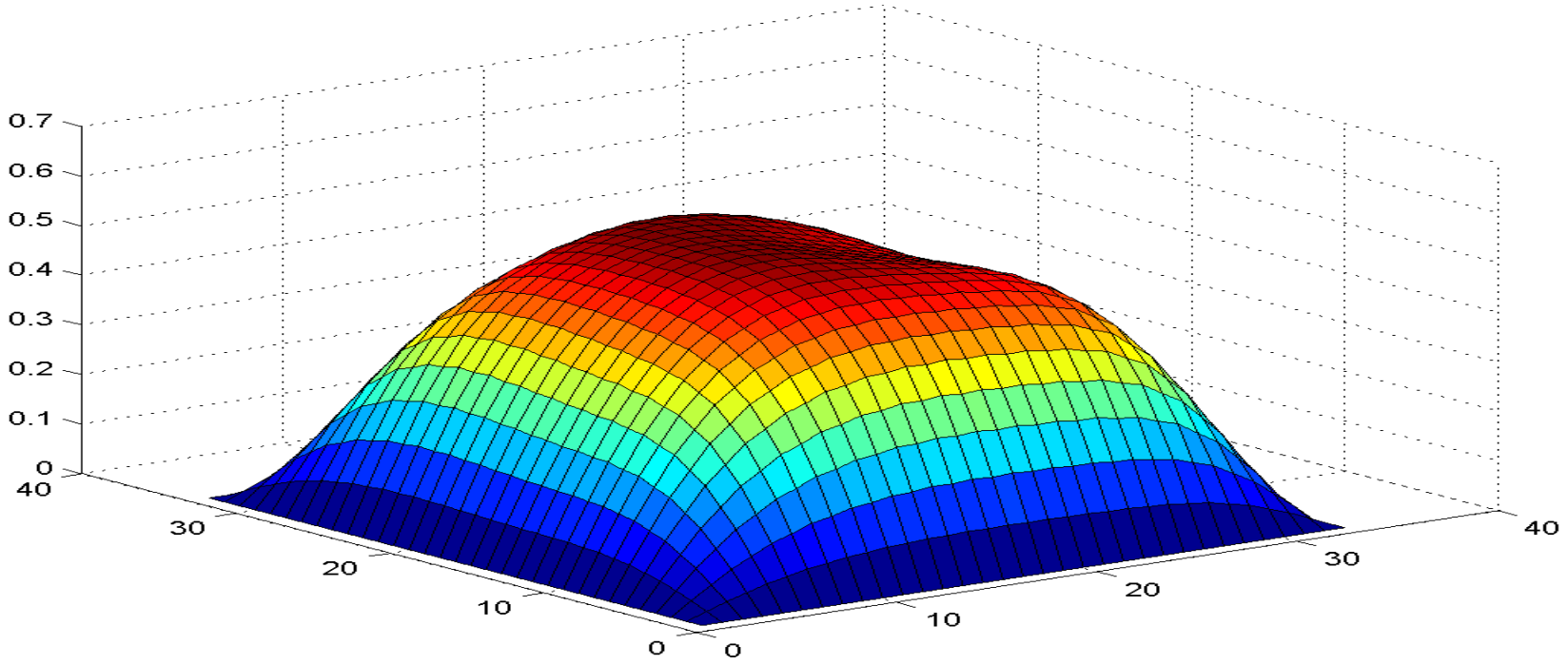

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

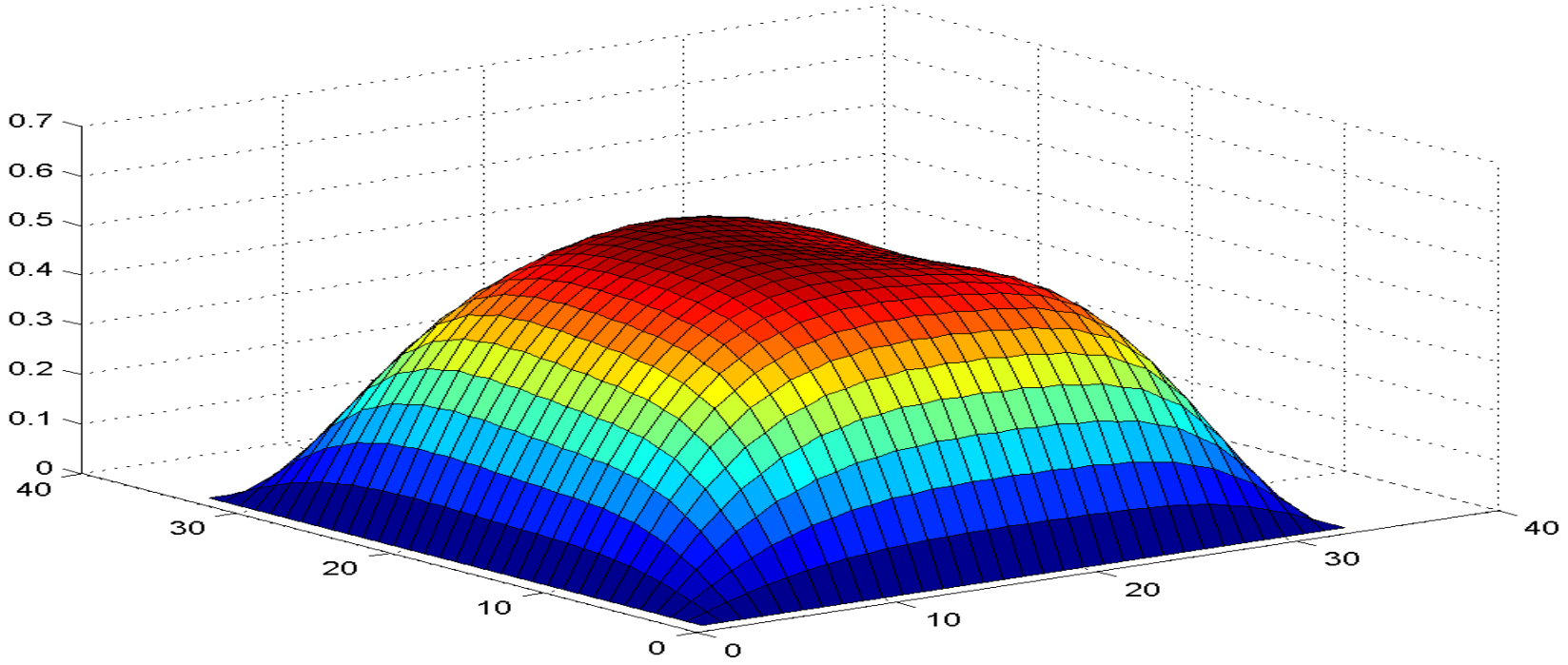

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40

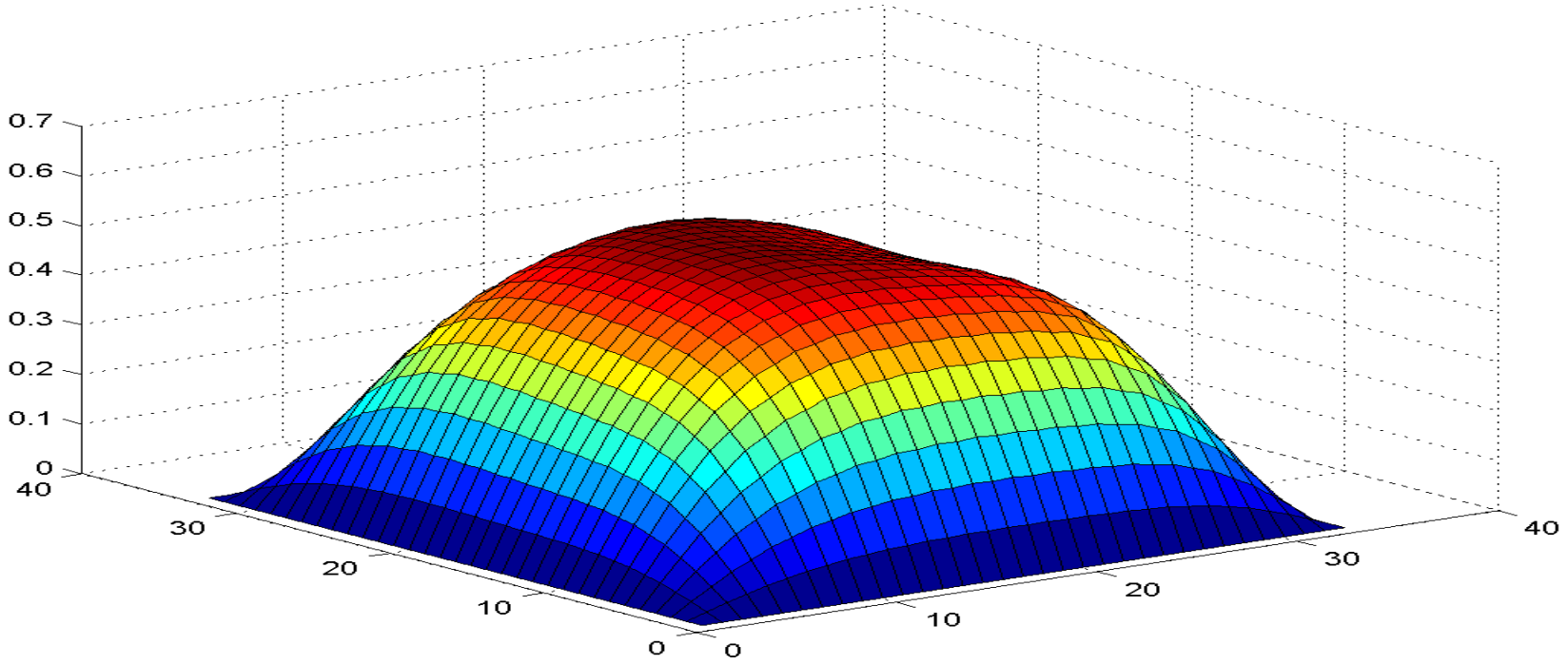

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
40
30
20
10
0
0
10
20
30
40


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:

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

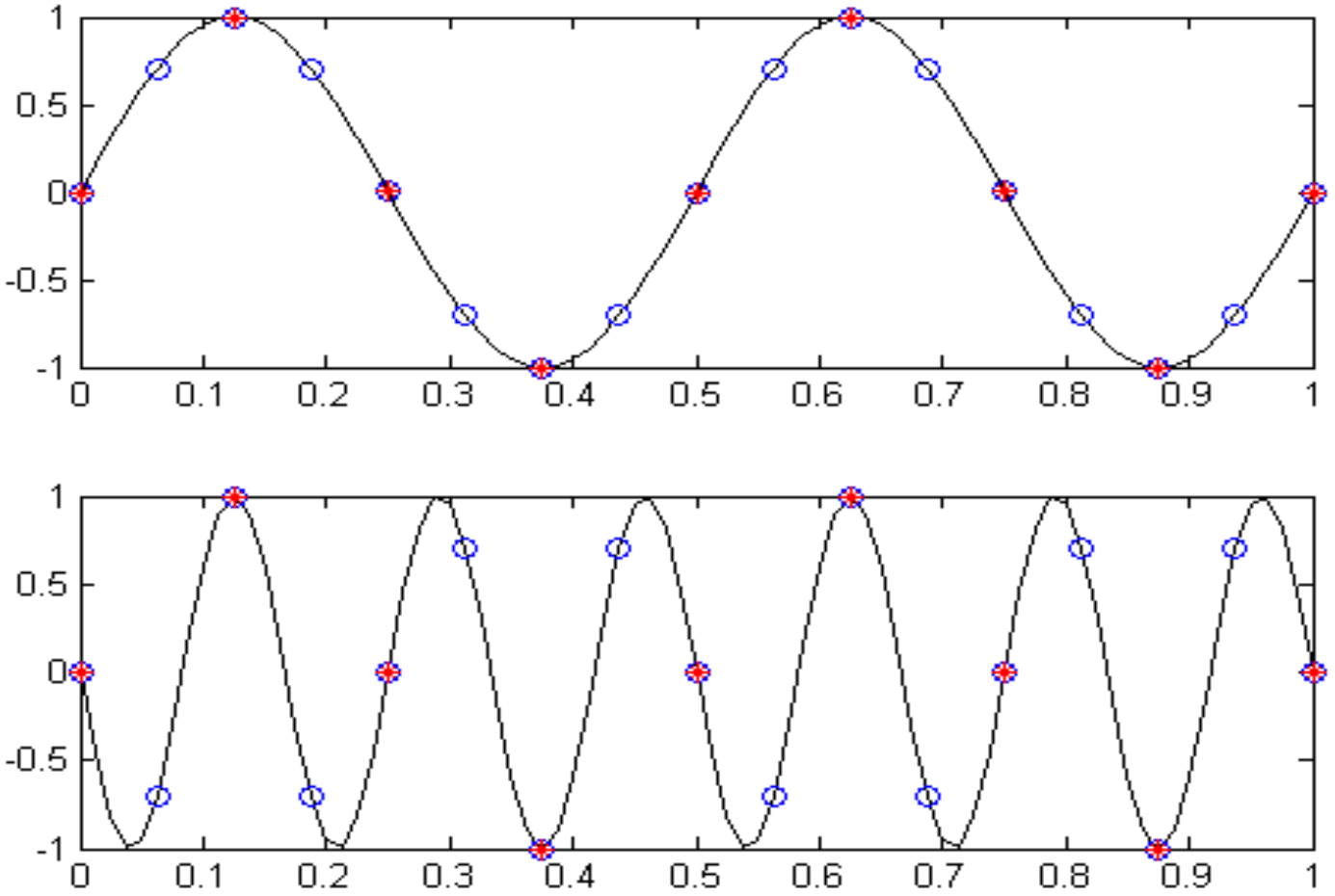
$$i = 1, \ldots N-1,$$

$$u^h_0 = u_0,$$

$$u^h_N = u_1.$$

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

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



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:

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

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.

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 $V(\nu_1, \nu_2)$

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

Relax on $L^h u^h = f^h$ ν_1 **times**

Set $f^{2h} = I_h^{2h}\left(f^h - L^h u^h\right), \quad u^{2h} = 0$

Relax on $L^{2h} u^{2h} = f^{2h}$ ν_1 **times**

Set $f^{4h} = I_{2h}^{4h}\left(f^{2h} - L^{2h} u^{2h}\right), \quad u^{4h} = 0$

Relax on $L^{4h} u^{4h} = f^{4h}$ ν_1 **times**

Set $f^{8h} = I_{4h}^{8h}\left(f^{4h} - L^{4h} u^{4h}\right), \quad u^{8h} = 0$

$\cdots$

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

$\cdots$

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

Relax on $L^{4h} u^{4h} = f^{4h}$ ν_2 **times**

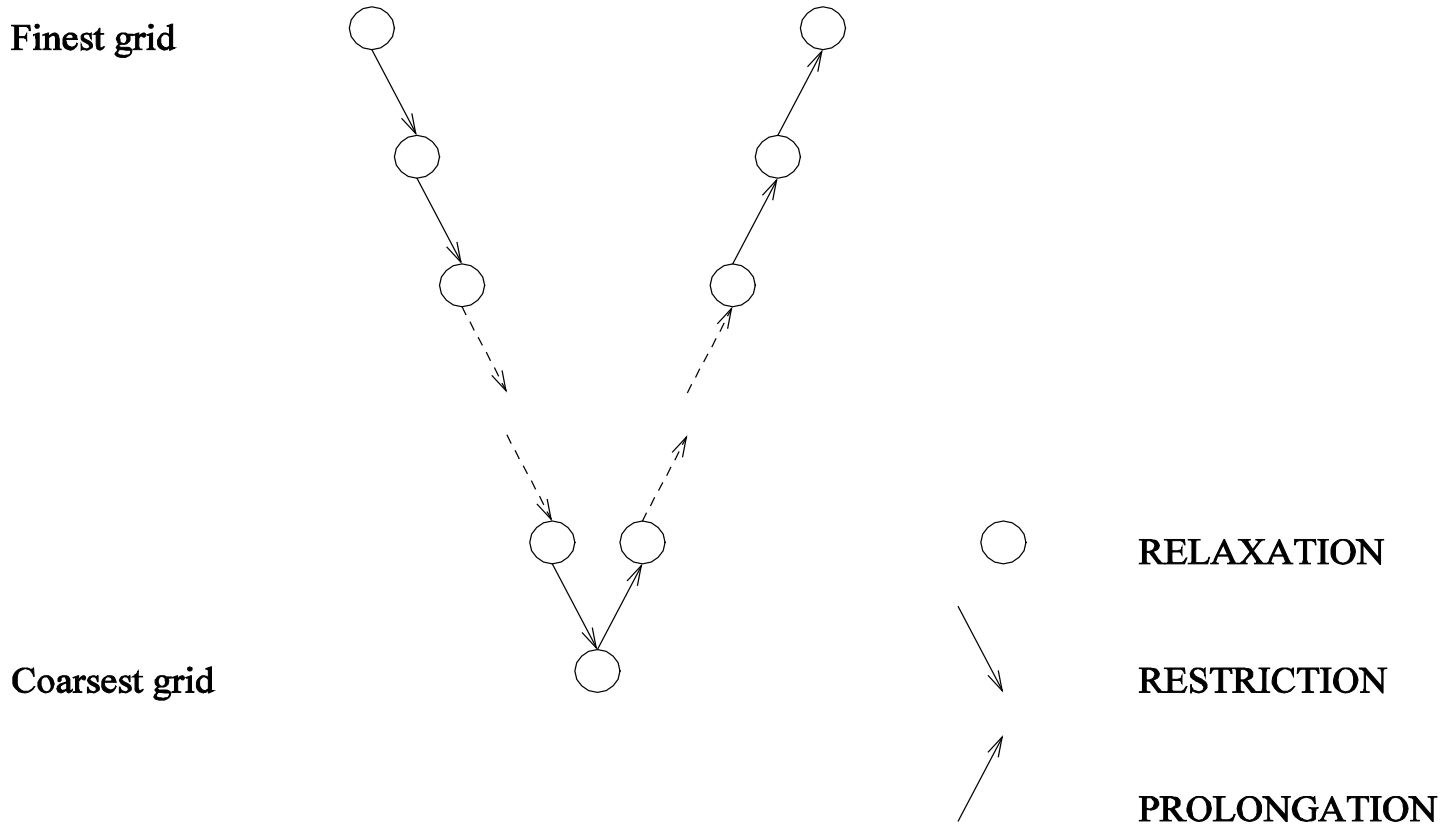
Correct $u^{2h} \leftarrow u^{2h} + I_{4h}^{2h} u^{4h}$

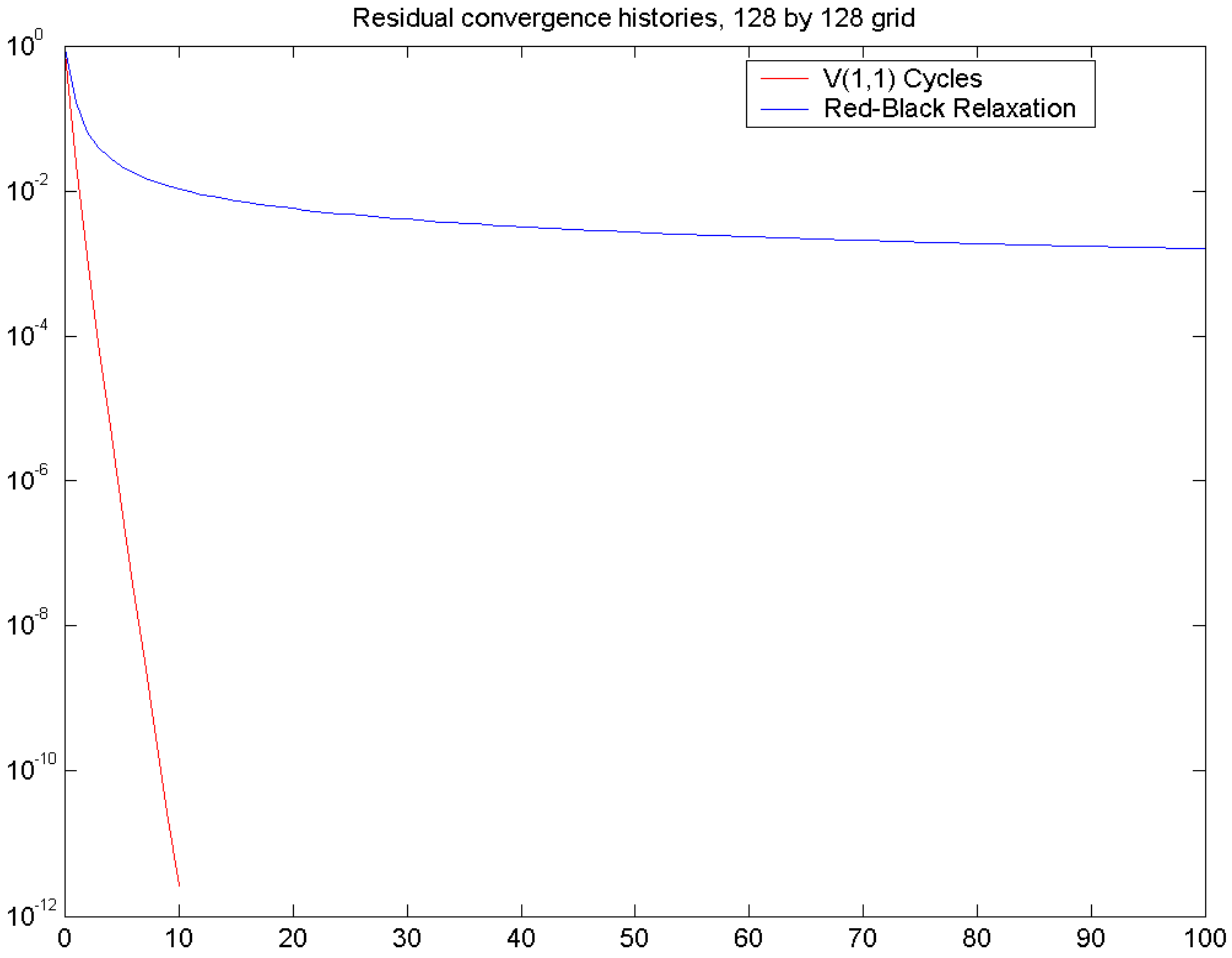
Relax on $L^{2h} u^{2h} = f^{2h}$ ν_2 **times**

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

Relax on $L^h u^h = f^h$ ν_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 γ, and define a γ–cycle recursively as follows: Relax; transfer to next coarser grid; perform γ γ–cycles; interpolate and correct; Relax. (On the coarsest grid define the γ–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 $\mathbf{S}^h$,

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

where $\mathbf{v}^h_{before}$ and $\mathbf{v}^h_{after}$ 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

ν_1 – # Relaxations before CGC

ν_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 $\mathbf{S}^h$ 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 (or an infinite grid). Also, we assume our operator, $\mathbf{L}^h$, to have constant coefficients. Hence, every element of the corresponding matrix, denoted by $\mathbf{A}$, satisfies:

$$\mathbf{A}_{i,j} = \mathbf{A}_{i-1,\, j-1 \,(\mathrm{mod}\, N)}.$$

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,

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

Here,

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

Proof.

By induction, for any integer p,

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

That is, ω^{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 λ, 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,

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

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

Let λ_k denote an eigenvalue of ω, 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,

$$\mathbf{v}_1^k = \lambda_k \mathbf{v}_0^k,$$
$$\mathbf{v}_2^k = \lambda_k \mathbf{v}_1^k = \lambda_k^2 \mathbf{v}_0^k,$$
$$\dots$$
$$\mathbf{v}_j^k = \lambda_k^j \mathbf{v}_0^k,$$
$$\dots$$
$$\mathbf{v}_0^k = \lambda_k \mathbf{v}_{N-1}^k = \lambda_k^N \mathbf{v}_0^k.$$

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

$$\lambda_k = 1^{1/N} = e^{i 2\pi k / N}.$$

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 2π. The grid-points are located at

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

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

The eigenvectors can be considered as grid-based functions (Fourier Components):

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

Here, k can be any integer, but note that φ_k aliases with φ_{k+N}. We therefore restrict:

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

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

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

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

The wavelength is $2\pi h/\theta$.

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)

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

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

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

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:

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

Examples (1D):

$$Lu = u_{xx},$$

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

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

Suppose we discretize L by

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

Then,

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

So,

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

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

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

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

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

Examples:

For 1D Poisson,

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

we obtain for Jacobi relaxation,

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

and for Gauss-Seidel relaxation,

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

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

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

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

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

The relaxation operator, R^h, is defined by

$$v^h_{after} = R^h v^h_{before}.$$

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

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

For damped Jacobi relaxation we have

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

So the symbol of damped Jacobi relaxation is

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

For Gauss-Seidel relaxation we have

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

Substituting the Fourier function yields

$$A_\theta e^{i\theta(j+1)} - 2\overline{A}_\theta e^{i\theta j} + \overline{A}_\theta e^{i\theta(j-1)} =$$

$$\left[A_\theta e^{i\theta} - 2\overline{A}_\theta + \overline{A}_\theta e^{-i\theta}\right] e^{i\theta j} = 0.$$

Hence,

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

The symbol of Gauss-Seidel relaxation is therefore

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

Aliasing Revisited

Note that

$$e^{i(\theta \pm 2\pi)x_j/h} = e^{i(\theta \pm 2\pi)j} =$$

$$e^{\pm i2\pi j} e^{i\theta j} = e^{i\theta j} = e^{i\theta x_j/h}.$$

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 ½ of the fine-grid frequencies – those in the range $|\theta| \le \pi/2$.

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

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

Example:

The smoothing factor of Gauss-Seidel relaxation is

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

$$= \max_{\pi/2 \le |\theta| \le \pi} \left| \frac{e^{i\theta}}{2 - e^{-i\theta}} \right|$$

$$= \max_{\pi/2 \le |\theta| \le \pi} \left| \frac{1}{2 - \cos(\theta) + i \sin(\theta)} \right|$$

$$= \max_{\pi/2 \le |\theta| \le \pi} \left| \frac{1}{\sqrt{\left(2 - \cos(\theta)\right)^2 + \sin^2(\theta)}} \right| = \frac{1}{\sqrt{5}}.$$

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

The smoothing factor of Jacobi relaxation with no damping is 1, because $\left|\cos\left(\pm\pi\right)\right| = 1$. 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 ω which minimizes μ of damped Jacobi? What is the corresponding μ?

Higher dimensions

The Fourier analysis can be generalized to d dimensions.

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

where

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

Higher dimensions

The Fourier components are

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

Where h_k is the mesh-size in the kth coordinate.

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

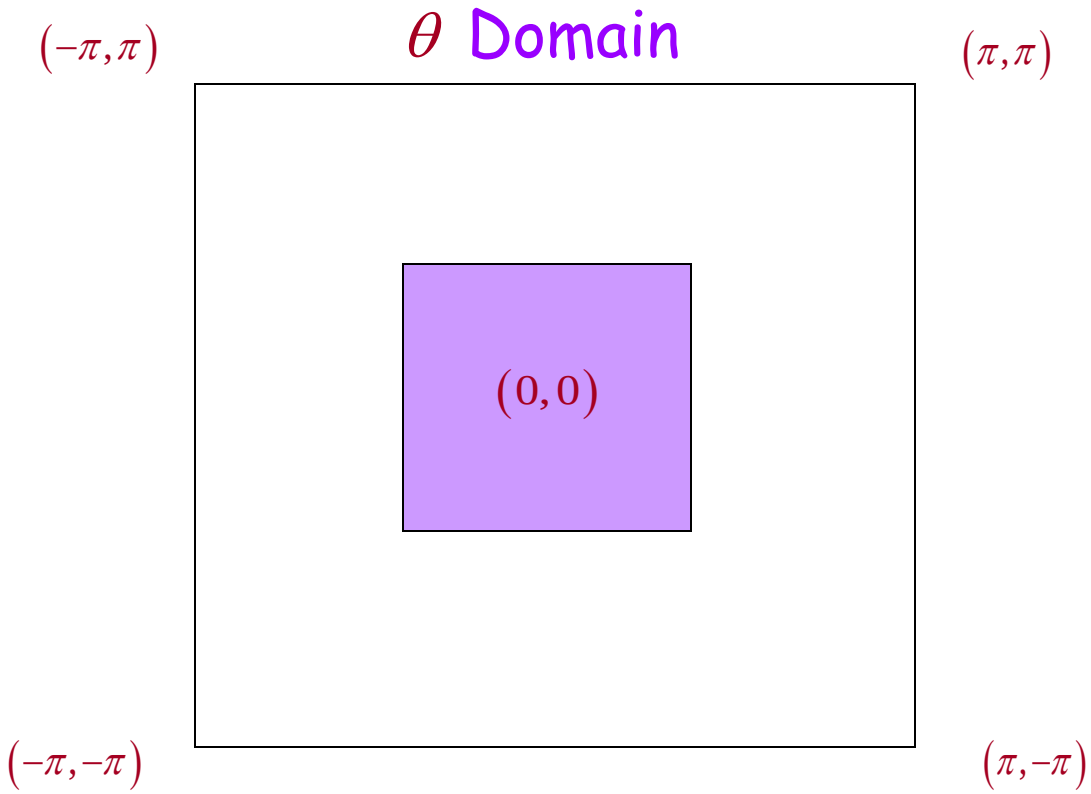
$$|\theta| \stackrel{def}{=} \max\left(|\theta_1|,\ldots,|\theta_d|\right).$$

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 μ is the maximum of $\left|\hat{R}(\theta)\right|$ over all θ for which the absolute value of at least one component is at least $\pi/2$.



The shaded area marks the part of the θ 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 ½.

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, μ. The input should be two small matrices representing the stencils of L^{h+} and L^{h-}, and also a damping parameter, ω. 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 θ'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

$$E^h\left(L^h\right) = \frac{\min\limits_{\pi/2 \le |\theta| \le \pi} \left|\hat{L}^h(\theta)\right|}{\max\limits_{\pi/2 \le |\theta| \le \pi} \left|\hat{L}^h(\theta)\right|},$$

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 .

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

Here,

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

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

Next, choose

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

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

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

Here,

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

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

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

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

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

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 ϕ be the angle between (s,t) and the grid-aligned coordinate system, (x,y). Hence,

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

with

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

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

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

Here,

$$\hat{L}^h = \frac{1}{h^2}\left[2\left(\varepsilon\ \cos(\theta_1)+\cos(\theta_2)-2(1+\varepsilon)\right)\right]$$

$$= -\frac{4}{h^2}\left[\varepsilon \sin^2(\theta_1/2)+\sin^2(\theta_2/2)\right] \ .$$

Setting $(\theta_1,\theta_2)=(\pi/2,0)$ vs. (π,π), for example, we obtain $E^h=O(\varepsilon),\quad \varepsilon\to 0,$ 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

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

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

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

Thus the smoothing factor is very poor for small ε 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 $\varepsilon \to 0$, 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

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

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

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

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.

Algebrization of Multigrid

There are many problems for which multigrid is suitable in principle but cannot be applied in a straightforward way. For example,

1. Unstructured grids and complex geometries

2. Non-PDE applications

Such situations require algebraic multigrid methods.

The multigrid components can be expressed as matrices. Consider, for example, the 1D model problem using linear interpolation and full-weighted residual transfers.

$$L^h = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & & & & \\ 1 & -2 & 1 & & & & & \\ & 1 & -2 & 1 & & & & \\ & & & \cdots & & & & \\ & & & 1 & -2 & 1 & & \\ & & & & 1 & -2 & 1 & \\ & & & & & 1 & -2 \end{bmatrix}$$

$$I_H^h = 2\left(I_h^H\right)^T = \frac{1}{2}\begin{bmatrix} 1 & & & & & & \\ 2 & & & & & & \\ 1 & 1 & & & & & \\ & 2 & & & & & \\ & 1 & 1 & & & & \\ & & & \cdots & & & \\ & & & & 1 & 1 & \\ & & & & & 2 & \\ & & & & & 1 & 1 \\ & & & & & & 2 \\ & & & & & & 1 \end{bmatrix}$$

Given the fine-grid matrix, L^h, and the prolongation and restriction matrices, I_H^h , and I_h^H, how should we define the coarse-grid matrix, L^H ?

The coarse grid should be able to correct smooth errors. We use the following (algebraic) definition of smoothness: An error v_{before}^h is smooth if it is in the range of interpolation. That is, if there exists some coarse-grid function, w^H, such that

$$v_{before}^h = I_H^h w^H .$$

The error after the coarse-grid correction is given by

$$v_{after}^{h} = C^{h} v_{before}^{h}$$

where

$$C^{h} = I^{h} - I_{H}^{h} \left(L^{H} \right)^{-1} I_{h}^{H} L^{h}.$$

Plugging in our smooth error we obtain:

$$v^h_{after} = C^h v^h_{before}$$

$$= \left[I^h - I^h_H \left(L^H \right)^{-1} I^H_h L^h \right] I^h_H w^H$$

$$= \left[I^h_H - I^h_H \left(L^H \right)^{-1} I^H_h L^h I^h_H \right] w^H$$

$$= I^h_H \left[I^H - \left(L^H \right)^{-1} I^H_h L^h I^h_H \right] w^H .$$

In order to annihilate the error we must choose the Petrov-Galerkin coarse-grid operator:

$$L^H = I^H_h L^h I^h_H .$$

For symmetric problems especially, the preferred choice for the restriction is the transpose of the prolongation. Along with the Galerkin coarse-grid operator this yields so-called variational coarsening, which arises naturally in finite-element formulations.

It remains only to define the prolongation (and, implicitly, the set of variables which defines the coarse grid). The prolongation operator should produce good approximate fine-grid values from given coarse-grid values. Therefore, I_H^h needs to be determined using L^h When used with appropriate coarse grids, such methods yield fast and robust algebraic solvers.

For tridiagonal matrices in 1D the different algebraic methods become the same: an exact multigrid solver that is equivalent to cyclic reduction

If the fine-grid equations are

$$a_i u_{i-1} + b_i u_i + c_i u_{i+1} = f_i \,,$$

$I = 1,\ldots, n-1$, with $a_1 = c_{n-1} \equiv 0$, we choose the prolongation matrix to be

$$I_H^h = \begin{bmatrix} -\frac{c_1}{b_1} & & & & & & \\ 1 & & & & & & \\ -\frac{a_3}{b_3} & -\frac{c_3}{b_3} & & & & & \\ & 1 & & & & & \\ & -\frac{a_5}{b_5} & & & & & \\ & & \cdots & & & & \\ & & & -\frac{c_{n-5}}{b_{n-5}} & & & \\ & & & 1 & & & \\ & & & -\frac{a_{n-3}}{b_{n-3}} & -\frac{c_{n-3}}{b_{n-3}} & & \\ & & & & 1 & & \\ & & & & -\frac{a_{n-1}}{b_{n-1}} & & \end{bmatrix}$$

Furthermore, we let $I_h^H = \left(I_H^h\right)^T$ and employ Galerkin coarsening. For smoothing we use half-Red-Black relaxation. That is, before restricting residuals we relax only on odd-indexed gridpoints, and after the coarse-grid correction only on even-indexed points.

Theorem: the two-level cycle is an exact solver. Furthermore, the coarse-grid equations are tridiagonal. Hence, the multigrid cycle is an exact solver.

Algebraic Multigrid (AMG)

Introduced by Brandt et al. (1983) and developed by Ruge and Stueben.

AMG takes the algebrization of multigrid to the limit. Here, a relaxation method is chosen (usually, point Gauss-Seidel), and then the coarse-grid variables are chosen by a heuristic graph algorithm such that each fine-grid variable depends strongly on one or more coarse-grid variable (i.e., with relatively large coefficient).

AMG enables us to handle unstructured and non-PDE problems.

An Abstract View of Algeraic Multigrid Methods

Consider the linear system

$$Au = f.$$

Suppose we partition the variables, u_i , into F variables and C variables, and permute the equations and variables to produce the following partitioning of the system:

$$Au = \begin{pmatrix} A_{FF} & A_{FC} \\ A_{CF} & A_{CC} \end{pmatrix} \begin{pmatrix} u_F \\ u_C \end{pmatrix} = \begin{pmatrix} f_F \\ f_C \end{pmatrix}.$$

An Abstract View of AMG

Given an approximate solution, $\tilde{u}$, define the error as

$$v = u - \tilde{u}.$$

Then, the partitioned equation for the error is

$$Av = \begin{pmatrix} A_{FF} & A_{FC} \\ A_{CF} & A_{CC} \end{pmatrix} \begin{pmatrix} v_F \\ v_C \end{pmatrix} = \begin{pmatrix} r_F \\ r_C \end{pmatrix},$$

where

$$r_F = f_F - A_{FF}\tilde{u}_F - A_{FC}\tilde{u}_C,$$
$$r_C = f_C - A_{CF}\tilde{u}_F - A_{CC}\tilde{u}_C.$$

An Abstract View of AMG

The upper block yields

$$A_{FF} v_F = r_F - A_{FC} v_C ,$$
$$\Rightarrow v_F = A_{FF}^{-1} \left(r_F - A_{FC} v_C \right).$$

Plugging this into the lower block yields

$$A_{CF} A_{FF}^{-1} \left(r_F - A_{FC} v_C \right) + A_{CC} v_C = r_C ,$$
$$\Rightarrow \left(A_{CC} - A_{CF} A_{FF}^{-1} A_{FC} \right) v_C = r_C - A_{CF} A_{FF}^{-1} r_F .$$

An Abstract View of AMG

Conclusion: the "ideal" prolongation and restriction are

$$P = \begin{pmatrix} -A_{FF}^{-1} A_{FC} \\ I \end{pmatrix}, \quad R = \begin{pmatrix} -A_{CF} A_{FF}^{-1} & I \end{pmatrix},$$

with the coarse-grid operator given by

$$A_C = RAP = A_{CC} - A_{CF} A_{FF}^{-1} A_{FC}.$$

An Abstract View of AMG

In particular, it is straightforward to verify that the two-level solution is exact in this case, provided that either a pre-relaxation or a post-relaxation eliminates r_F.

(If this is done by post-relaxation, only u_F should be relaxed.)

The problem: A_{FF}^{-1} is not sparse, and therefore, neither are P and R. Therefore, we generally look for good sparse approximations.

One exception is tri-diagonal systems, where A_{FF} is diagonal. In this case the multigrid V-cycle with the appropriate prolongation and restriction, and with relaxation only on u_F is an exact solver, equivalent to total reduction.