Iterative Solution of Sparse Equation Systems

Stefan Lang

Interdisciplinary Center for Scientific Computing (IWR)
University of Heidelberg
INF 368, Room 532
D-69120 Heidelberg
phone: 06221/54-8264
email: Stefan.Lang@iwr.uni-heidelberg.de

WS 15/16

Topics

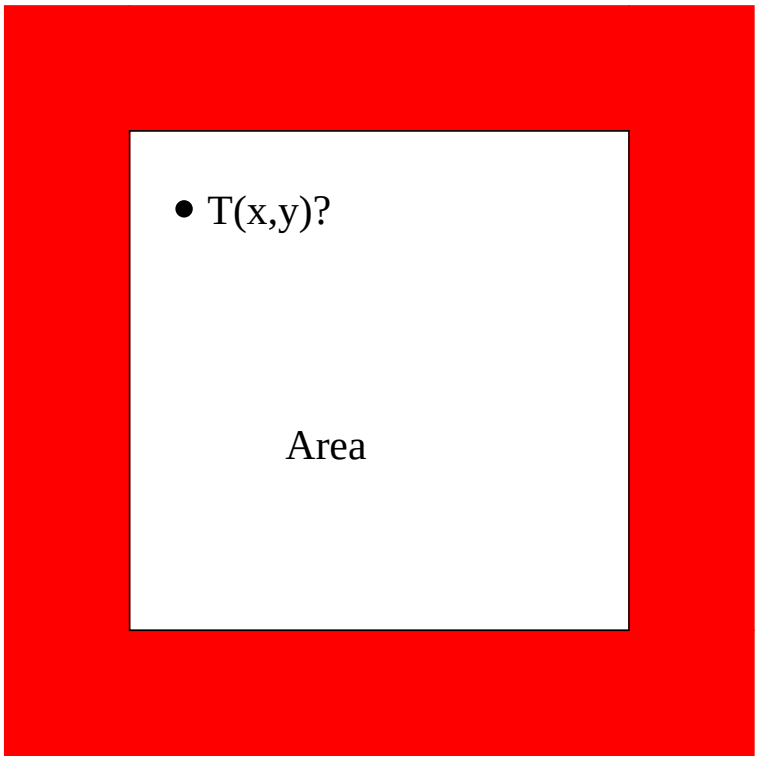
Iterative Solution of Sparse Linear Equation Systems

- Problem formulation
- Iteration methods
- Parallelisation
- Multigrid methods
- Parallelisation of multigrid methods

Problem Formulation: Example

A continuous problem and its discretisation:

- Example: A thin, quadratic metal plate is fixed at every side.
- The temporal constant temperature distribution at the boundary of the metal plate is known.
- Which temperature exists at each inner point of the metal plate, if the system is in a stationary state?



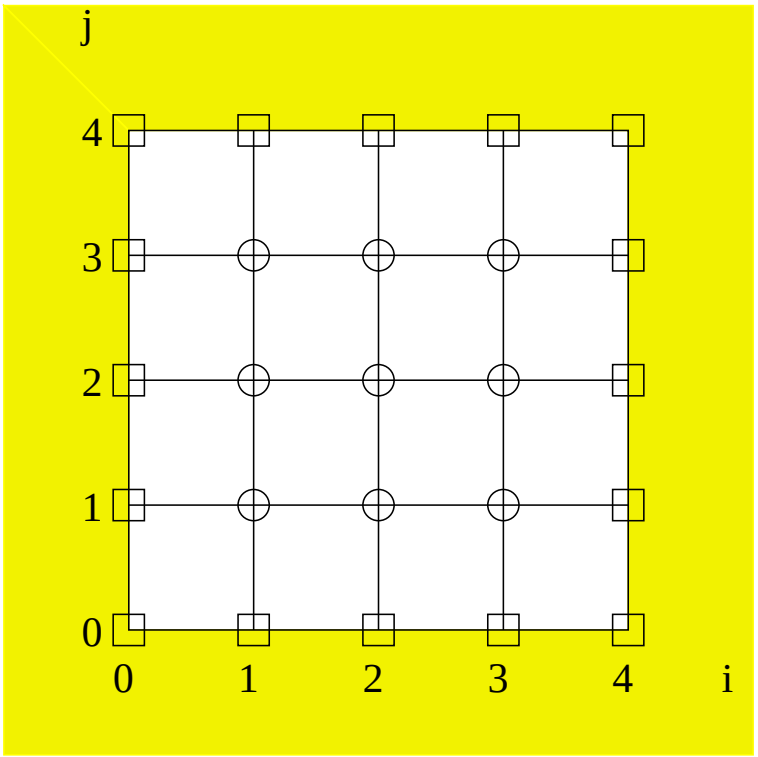
Problem Formulation: Continuous

- The process of heat conduction can be described (approximately) by a mathematical model.
- The geometry of the metal plate is decribed by an area $\Omega \subset \mathcal{R}^2$.
- Wanted is the temperature distribution $T(x, y)$ for all $(x, y) \in \Omega$.
- The temperature $T(x, y)$ for (x, y) on the boundary $\partial\Omega$ is known.
- Is the metal plate homogeneous (same heat conduction coefficient everywhere), then the temperature in the inner domain is described by the partial differential equation

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0\,, \quad T(x, y) = g(x, y) \text{ auf } \partial\Omega. \tag{1}$$

Problem Formulation: Discrete

- In the computer one cannot determine the temperature at every position $(x, y) \in \Omega$ (innumerable many), but only at some selected ones.
- For that be $\Omega = [0, 1]^2$ choosen specially (unit square).
- Via the parameter $h = 1/N$, for a $N \in \mathbf{N}$, we choose specially the points $(x_i, y_j) = (ih, jh)$, for all $0 \leq i, j \leq N$.
- One denotes this set of points also as regular, equidistant grid.



The points at the boundary have been marked with other symbols (squares) as the inner ones (circles).

Discretisation I

How can the temperature T_{ij} at point (x_i, y_j) be determined?

- By a standard method: the method of „Finite Differences“
- Idea: The temperature at point (x_i, y_j) is expressed by the values of the four neighboring points:

$$T_{i,j} = \frac{T_{i-1,j} + T_{i+1,j} + T_{i,j-1} + T_{i,j+1}}{4} \tag{2}$$

$$\iff \quad T_{i-1,j} + T_{i+1,j} - 4T_{i,j} + T_{i,j-1} + T_{i,j+1} = 0 \tag{3}$$

für $1 \leq i, j \leq N - 1$.

- From the form of (3) one recognices, that all $(N-1)^2$ equations for $1 \leq i, j \leq N - 1$ together form a linear equation system:

$$AT = b \tag{4}$$

Discretisation II

- Here $G(A)$ corresponds exactly to the above drawn grid, if one neglects the boundary points (squares). The righthand side b of (3) is not even zero, but contains the temperature values at the boundary!
- The such calculated temperature values $T_{i,j}$ at the points (x_i, y_j) are *not* identical with the solution $T(x_i, y_i)$ of the partial differential equation (1). Furthermore applies

$$|T_{i,j} - T(x_i, y_i)| \leq O(h^2) \tag{5}$$

- This error is denoted as „discretisation error“. An increase in the size of N corresponds such to an exacter temperature calculation.

Iteration Methods I

- We now want to solve the equation system (4) „iteratively“. Herefore we determine an arbitrary value of the temperature $T_{i,j}^0$ at each point $1 \leq i,j \leq N-1$ (the temperature at the boundary is wellknown).
- Starting from this approximate solution we now want to calculate an improved solution. Herefore we use the formula (2) and set

$$T_{i,j}^{n+1} = \frac{T_{i-1,j}^n + T_{i+1,j}^n + T_{i,j-1}^n + T_{i,j+1}^n}{4} \quad \text{für alle } 1 \leq i,j \leq N-1. \qquad (6)$$

- Obviously the improved values $T_{i,j}^{n+1}$ can be calculated simultaneously for each of the indices (i,j), since they only depend on the old values $T_{i,j}^n$.

Iteration Methods II

- One can indeed show, that

$$\lim_{n \to \infty} T_{i,j}^n = T_{i,j} \tag{7}$$

  applies.
- The error $|T_{i,j}^n - T_{i,j}|$ in the n-th approximate solution is denoted as „iteration error“.
- How large is this iteration error then? One needs a criterium up to which n one needs to compute.

Iteration Methods III

- Herefore one considers how well the values $T_{i,j}^n$ fulfill the equation (3), this means we set

$$E^n = \max_{1 \leq i,j \leq N-1} \left| T_{i-1,j}^n + T_{i+1,j}^n - 4T_{i,j}^n + T_{i,j-1}^n + T_{i,j+1}^n \right|$$

- Commonly one uses this error only relatively, thus one iterates as long until

$$E^n < \epsilon E^0$$

applies. Then the initial error E^0 has been reduced by the reduction factor ϵ.

- This leads us to the sequential method:

choose N, ϵ;
choose $T_{i,j}^0$;
$E^0 = \max_{1 \leq i,j \leq N-1} \left| T_{i-1,j}^0 + T_{i+1,j}^0 - 4T_{i,j}^0 + T_{i,j-1}^0 + T_{i,j+1}^0 \right|$;
$n = 0$;
while $(E^n \geq \epsilon E^0)$
{
  for $(1 \leq i,j \leq N-1)$
    $T_{i,j}^{n+1} = \frac{T_{i-1,j}^n + T_{i+1,j}^n + T_{i,j-1}^n + T_{i,j+1}^n}{4}$;
  $E^{n+1} = \max_{1 \leq i,j \leq N-1} \left| T_{i-1,j}^{n+1} + T_{i+1,j}^{n+1} - 4T_{i,j}^{n+1} + T_{i,j-1}^{n+1} + T_{i,j+1}^{n+1} \right|$;
  $n = n + 1$;
}

Iteration Methods IV

- All that can be written more compact in vector notation. Then $AT = b$ is the equation system (4) to be solved. The approximation values $T_{i,j}^n$ correspond to vectors T^n each.
- Formally T^{n+1} is calculated as

$$T^{n+1} = T^n + D^{-1}(b - AT^n)$$

with the diagonal matrix $D = diag(A)$. This scheme is denoted as Jacobi method.

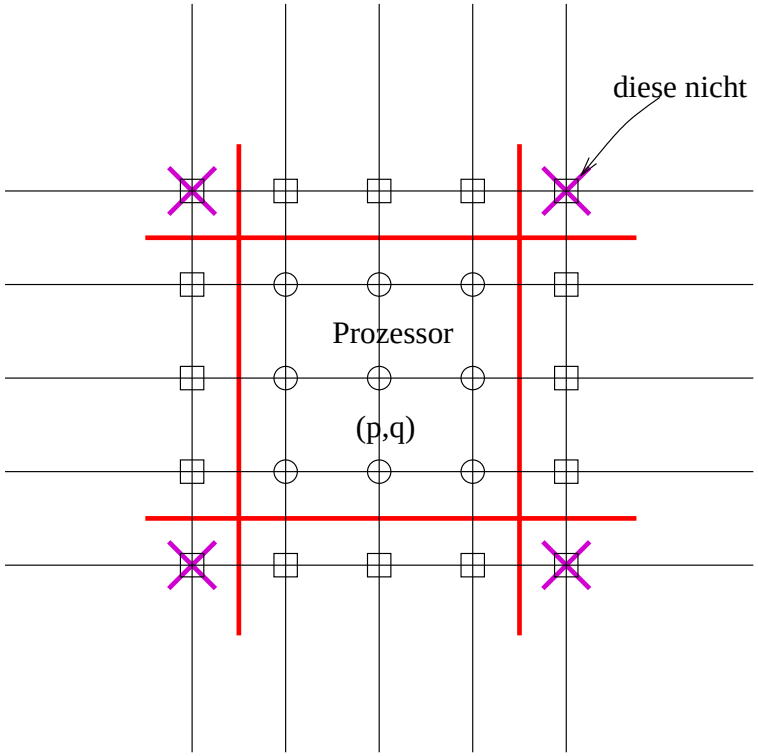
- The error E^n is constituted by

$$E^n = \|b - A \cdot T^n\|_\infty ,$$

where $\|\cdot\|_\infty$ is the maximum norm of a vector.

Parallelisation I

- The algorithm allows again a data parallel formulation.
- Therefore the $(N+1)^2$ grid points are subdivided onto a $\sqrt{P} \times \sqrt{P}$ processor array by partitioning of the index set $I = \{0, \ldots, N\}$.
- The partitioning happens here *block-wise*:



Parallelisation II

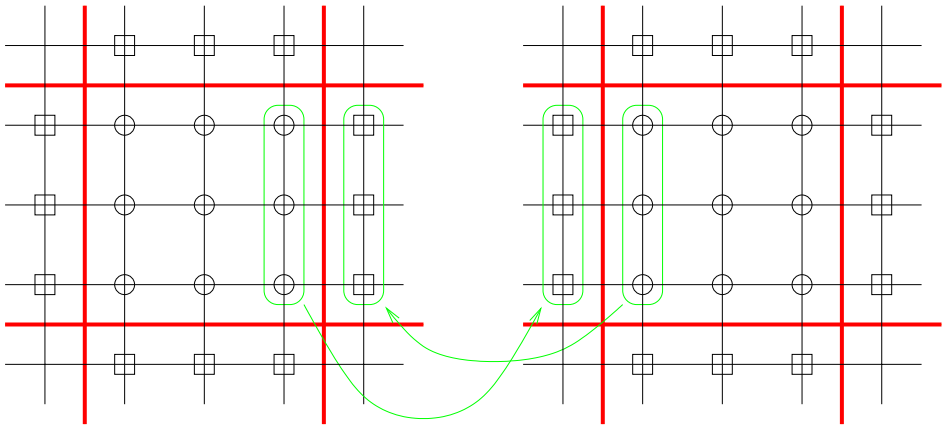
- Processor (p,q) computes then the values $T_{i,j}^{n+1}$ with $(i,j) \in \{start(p),\ldots,end(p)\} \times \{start(q),\ldots,end(q)\}$.
- To do this, he needs however also the values $T_{i,j}^{n}$ from the neighboring processors with $(i,j) \in \{start(p)-1,\ldots,end(p)+1\} \times \{start(q)-1,\ldots,end(q)+1\}$.
- These are the nodes, that have been marked with squares in the figure above!
- Each processor stores such beyond its assigned grid points an additional layer of grid points.

Parallelisation III

- The parallel algorithm therefore consists of the following steps:

  initial values $T_{i,j}^0$ are known by all processors.
  while $(E^n > \epsilon E^0)$
  {
  calculate $T_{i,j}^{n+1}$ for $(i,j) \in \{start(p), \ldots, end(p)\} \times \{start(q), \ldots, end(q)\}$;
  exchange boundary values (squares) with neighbors;
  calculate E^{n+1} for $(i,j) \in \{start(p), \ldots, end(p)\} \times \{start(q), \ldots, end(q)\}$;
  Determine global maximum;
  $n = n + 1$;
  }

- In the exchange step two neighboring processors exchange values:



Parallelisation IV

- For this exchange step one uses either asynchronous communication or synchronous communication with coloration.
- We calculate the scalability of a *single* iteration:

$$W = T_S(N) = N^2 t_{op} \quad \Longrightarrow N = \sqrt{\frac{W}{t_{op}}}$$

$$T_P(N,P) = \underbrace{\left(\frac{N}{\sqrt{P}}\right)^2 t_{op}}_{\text{calculation}} + \underbrace{\left(t_s + t_h + t_w \frac{N}{\sqrt{P}}\right) 4}_{\text{boundary exchange}} + \underbrace{(t_s + t_h + t_w) \operatorname{ld} P}_{\substack{\text{global} \\ \text{comm.: max.} \\ \text{for } E^n}}$$

$$T_P(W,P) = \frac{W}{P} + \frac{\sqrt{W}}{\sqrt{P}} \frac{4t_w}{\sqrt{t_{op}}} + (t_s + t_h + t_w) \operatorname{ld} P + 4(t_s + t_h)$$

$$T_O(W,P) = PT_P - W =$$

$$= \sqrt{W}\sqrt{P} \frac{4t_w}{\sqrt{t_{op}}} + P \operatorname{ld} P (t_s + t_h + t_w) + P4(t_s + t_h)$$

Parallelisation V

- Asymptotically we obtain the iso-efficiency function $W = O(P \operatorname{ld} P)$ from the second term, albeit the first term will be dominant for practical values of N dominant. The algorithm is nearly optimal scalable.
- Because of the block-wise partitioning one has a surface-to-volume effect: $\frac{N}{\sqrt{P}} \Big/ \left(\frac{N}{\sqrt{P}}\right)^2 = \frac{\sqrt{P}}{N}$. In three space dimensions one obtains $\left(\frac{N}{P^{1/3}}\right)^2 \Big/ \left(\frac{N}{P^{1/3}}\right)^3 = \frac{P^{1/3}}{N}$.
- For same N and P the efficiency is such a little bit worse compared to two dimensions.

Multigrid Methods I

- If we ask about the total efficiency of a method, then the number of operations is distinctive.
- Hereby is

$$T_S(N) = IT(N) \cdot T_{IT}(N)$$

- How many iterationen have now indeed to be executed depends besides N of course on the used method.
- For that one obtains the following classifications:

$$\begin{aligned}
\text{Jacobi, Gauß-Seidel}: &\quad IT(N) = \mathcal{O}(N^2) \\
\text{SOR with } \omega_{\text{opt}}: &\quad IT(N) = \mathcal{O}(N) \\
\text{Conjugated gradients (CG)}: &\quad IT(N) = \mathcal{O}(N) \\
\text{Hierarchical basis d=2}: &\quad IT(N) = \mathcal{O}(\log N) \\
\text{Multigrid methods}: &\quad IT(N) = \mathcal{O}(1)
\end{aligned}$$

- The time for an iteration $T_{IT}(N)$ is there for all schemes in $\mathcal{O}(N^d)$ with comparable constant (in the region of 1 to 5).

Multigrid Methods II

- We such see, that e.g. the multigrid method is much faster than the Jacobi method.
- Also the parallelisation of the Jacobi method does not help, since it applies:

$$T_{P,\text{Jacobi}}(N,P) = \frac{\mathcal{O}(N^{d+2})}{P} \quad \text{und} \quad T_{S,\text{MG}}(N) = \mathcal{O}(N^d)$$

- A doubling of N results in a fourfold increase of the effort for the parallelised Jacobi scheme in comparision to the `sequential` multigrid method!
- This leads to a fundamental paradigm of parallel programming:

> Parallelise the best sequential algorithm, if possible anyhow!

Multigrid Methods III

- Let us consider again the discretisation of the Laplace equation $\Delta T = 0$.
- This leads to the linear equation system

$$Ax = b$$

- Here the vector b is determined by the Dirichlet boundary values. Now be an approximation of the solution given by x^i. Herefore set the iteration error

$$e^i = x - x^i$$

- Because of the linearity of A we can conclude the following:

$$Ae^i = \underbrace{Ax}_{b} - Ax^i = b - Ax^i =: d^i$$

- Here we call d^i the `defect` .
- A good approximation for e^i is calculated by the solution of

$$Mv^i = d^i \quad \text{also} \quad v^i = M^{-1}d^i$$

- Herefore be M easier to solve than A (in $\mathcal{O}$(N) steps, if $x \in \mathbb{R}^N$).

Multigrid Methods IV

- For special M we get the already known iteration method :

$$\begin{aligned} M &= I && \rightarrow \text{Richardson} \\ M &= \text{diag}(A) && \rightarrow \text{Jacobi} \\ M &= L(A) && \rightarrow \text{Gauß-Seidel} \end{aligned}$$

- We obtain the linear iteration method of the form

$$x^{i+1} = x^i + \omega M^{-1}(b - Ax^i)$$

- Here the $\omega \in [0, 1]$ is a damping factor .
- For the error $e^{i+1} = x - x^{i+1}$ applies:

$$e^{i+1} = (I - \omega M^{-1}A)e^i$$

- Here we denote the iteration matrix $I - \omega M^{-1}A$ with S.
- The scheme is exactly then convergent if applies ($\lim_{i\rightarrow\infty} e^i = 0$). This holds if the largest absolut eigen value of S is smaller than one.

Smoothing Property I

- If the matrix A is symmetric and positive definite, then it has only real, positive eigen values λ_k for eigen vectors z_k.
- The Richardson iteration

$$x^{i+1} = x^i + \omega(b - Ax^i)$$

leads because of M = I to error

$$e^{i+1} = (I - \omega A)e^i$$

- Now we set the damping factor $\omega = \frac{1}{\lambda_{\max}}$ and consider $e^i = z_k (\forall k)$.
- Then we obtain

$$e^{i+1} = \left(I - \frac{1}{\lambda_{\max}} A\right) z_k = z_k - \frac{\lambda_k}{\lambda_{\max}} z_k = \left(1 - \frac{\lambda_k}{\lambda_{\max}}\right) e^i$$

$$\left(1 - \frac{\lambda_k}{\lambda_{\max}}\right) = \begin{cases} 0 & \lambda_k = \lambda_{\max} \\ \approx 1 & \lambda_k \text{ small } (\lambda_{\min}) \end{cases}$$

Smoothing Property II

- In the case of small eigenvalues we thus have a bad damping of the error (it has the order of magnitude of 1 - $\mathcal{O}$ (h^2)).
- This behaviour is qualitatively identical for the Jacobi and Gauß-Seidel iteration methods.
- But, to small eigenvalues belong long-wave eigenfunctions.
- This long-wave errors are such damped only very badly.
- With a pictorial view the iteration methods only offer a local smoothing of the error on which they work, since they get new iteration values only from values in the local neighborhood.
- Fast oscillations could be smoothed out fast, whilst long-wave errors survive the local smoothing operations quite unmodified.

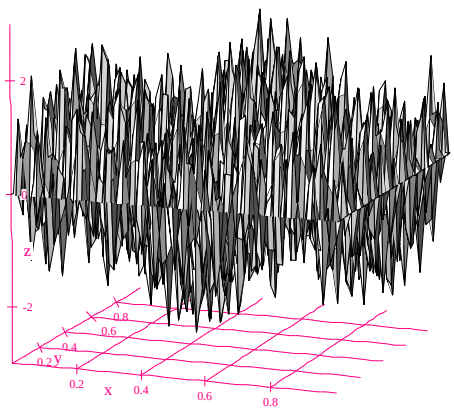
Smoothing Property III

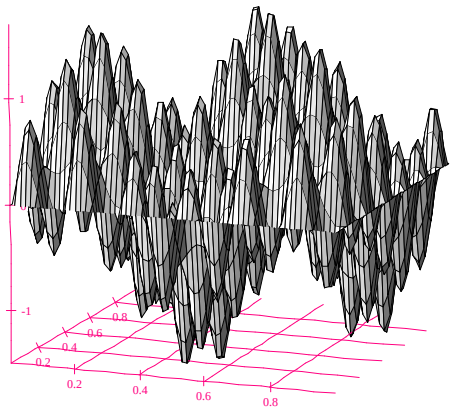
For illustration purposes we consider the following example:

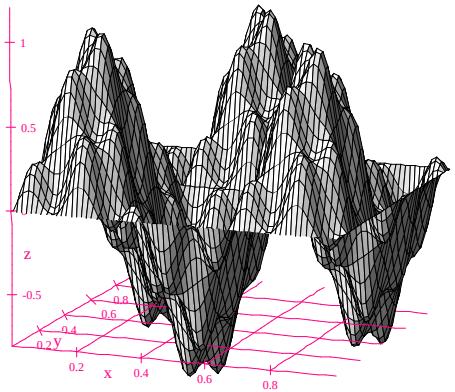
The Laplacian equation $-\Delta u = f$ is discretized via a five-point stencil on a structured grid. The associated eigenfunctions are $\sin(\nu\pi x)\sin(\mu\pi y)$*, where* $1 \leq \nu$ *and* $\mu \leq h^{-1} - 1$ *apply. We set* $h = \frac{1}{32}$ *and the initial error to*

$$e^0 = \sin(3\pi x)\sin(2\pi y) + \sin(12\pi x)\sin(12\pi y) + \sin(31\pi x)\sin(31\pi y).$$

With $\omega = \frac{1}{\lambda_{max}}$ *one obtains the damping factors (per iteration) for the Richardson iteration as 0.984, 0.691 and 0 for the individual eigenfunctions. The graphs below show the initial error and the error after one resp. five iterations.*

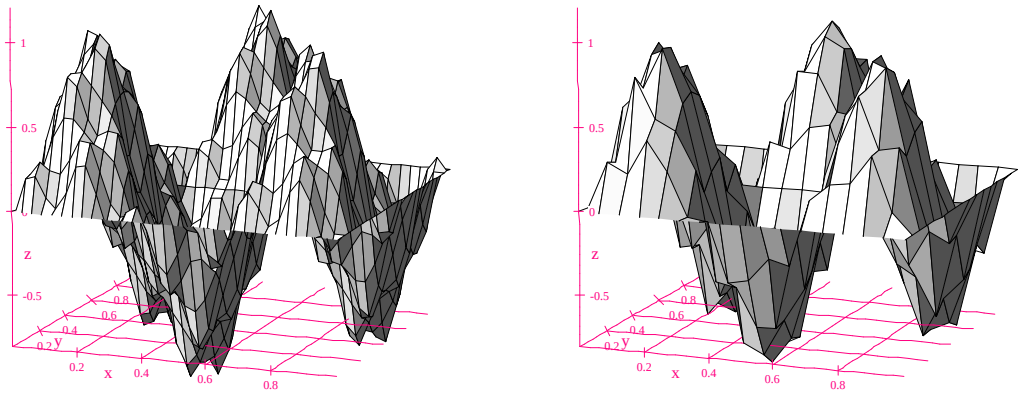






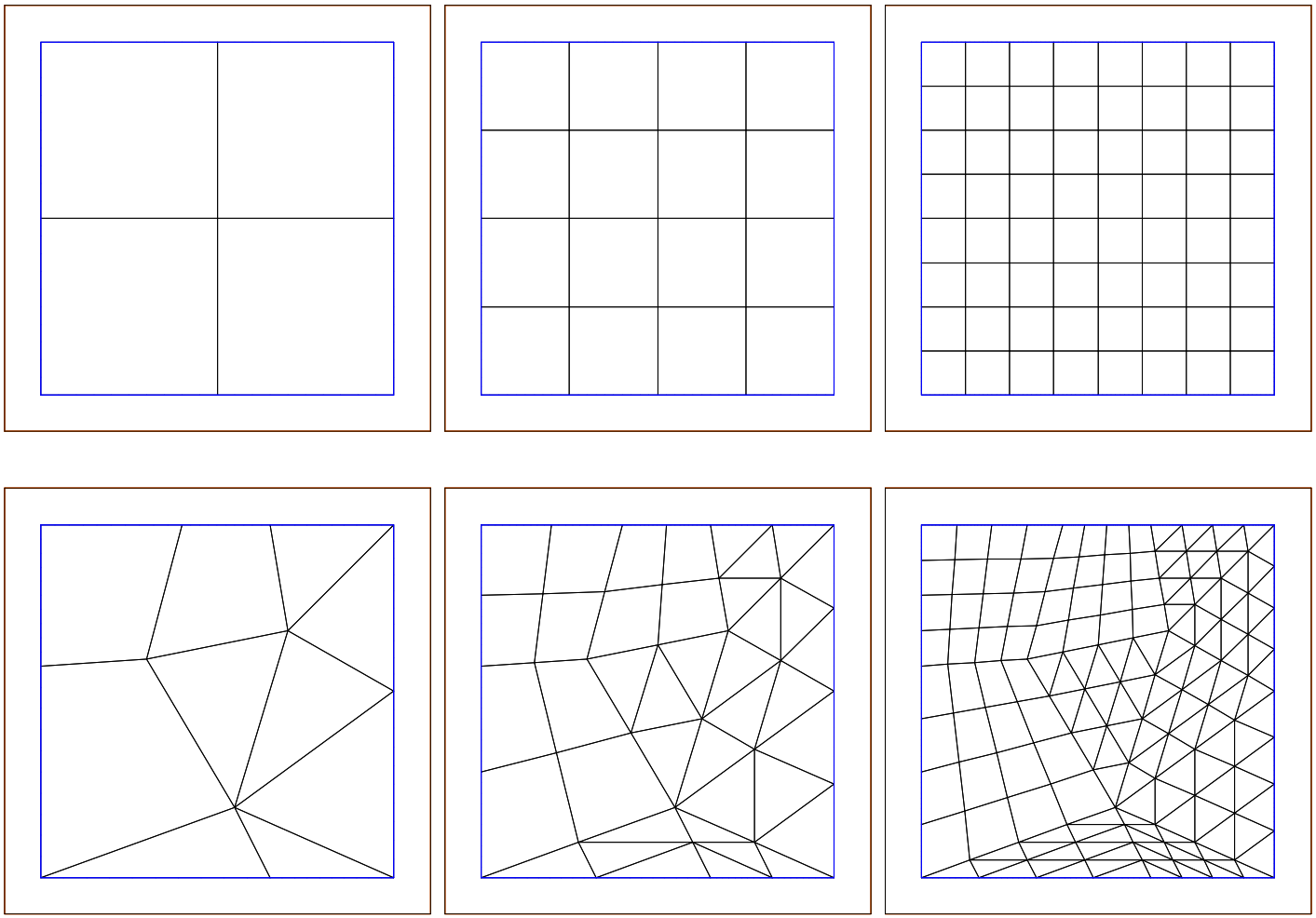
Smoothing Property IV

- From this the idea arises to represent the long wave error on coarser grids, after smoothing out the fast oscillations.
- On this coarser grids the effort is then smaller to smooth the error curve.
- Because the curve is somehow smooth after presmoothing, this restriction onto fewer grid points is well possible.



Grid Hierarchy I

- One constructs thus a complete sequence of grids of different accuracy.
- At first one smoothes out on the finest grid the short wave error functions.
- Then we restrict to the next coarser grid, and so on.



Grid Hierarchy II

- According to that one obtains a complete sequence of linear systems

$$A_l x_l = b_l,$$

  since the number of grid points N and therefore the length of x decreases on coarser grids.
- Of couse one wants to return after this restriction again to the original fine grid.
- Herefore we perform a coarse grid correction.
- Let us assume we are on grid level l, thus we consider the LES

$$A_l x_l = b_l$$

- On this level the iterate x_l^i is given with an error of e_l^i, thus the error equation

$$A e_l^i = b_l - A_l x_l^i$$

  Lets suppose, x_l^i is the result of ν_1 Jacobi, Gauß-Seidel or similar iterations.
- Then e_l^i is relatively smooth and thus also properly representable on a coarser grid, this means it can be interpolated well from a coarser grid to a finer one.

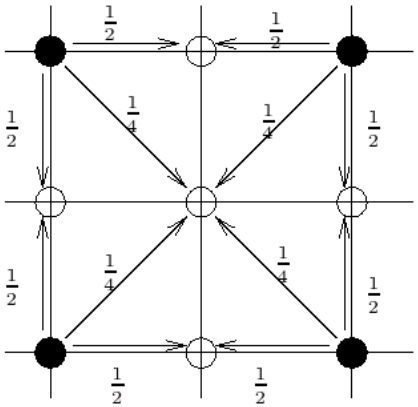
Grid Hierarchy III

- For this be v_{l-1} the error on the coarser grid.
- Then with good approximation applies

$$e_l^i \approx P_l v_{l-1}$$

- Here P_l is an interpolation matrix (prolongation), that performs a linear interpolation and changes the coarse grid vector into a fine grid vector.

1	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	0	0
0	1	0	0
$\frac{1}{2}$	0	$\frac{1}{2}$	0
$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$
0	$\frac{1}{2}$	0	$\frac{1}{2}$
0	0	1	0
0	0	$\frac{1}{2}$	$\frac{1}{2}$
0	0	0	1



Two-grid and Multigrid Methods I

- Through combination of the equations above one obtains the equation for v_{l-1} by

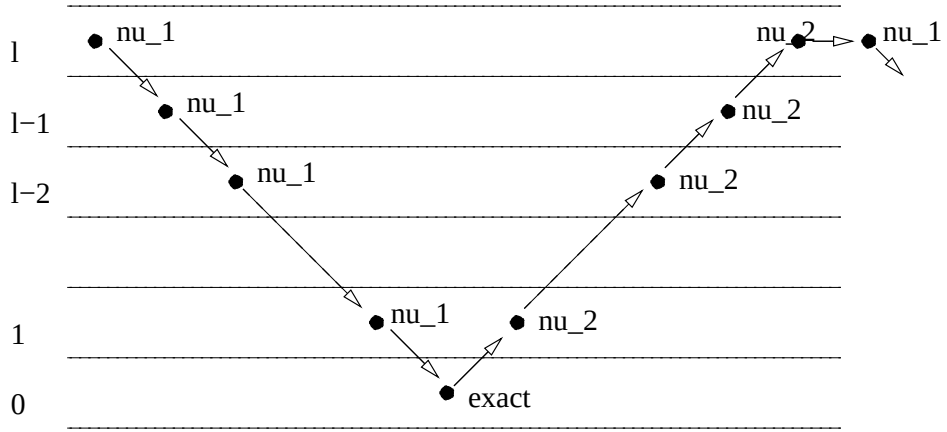
$$R_l A P_l v_{l-1} = R_l (b_l - A_l x_l^i)$$

- Here is $R_l A P_l =: A_{l-1} \in \mathbb{R}^{N_{l-1} \times N_{l-1}}$ and R_l is the restriction matrix, for that one takes e.g. $R_l = P_l^T$.
- The so called two-grid method consists now of the two steps:
  1. ν_1 Jacobi iterations (on level l)
  2. coarse grid correction $x_l^{i+1} = x_l^i + P_l A_{l-1}^{-1} R_l (b_l - A_l x_l^i)$
- The recursive application leads to the multigrid methods.

mgc(l, x_l, b_l)
{
  if (l == 0) $x_0 = A_0^{-1} b_0$;
  else {
    ν_1 iterations one-grid method on $A_l x_l = b_l$; // presmoothing
    $d_{l-1} = R_l (b_l - A_l x_l)$;
    $v_{l-1} = 0$;
    for ($g = 1, \ldots, \gamma$)
      mgc($l - 1, v_{l-1}, d_{l-1}$);
    $x_l = x_l + P_l v_{l-1}$;
    ν_2 iterations one-grid method on $A_l x_l = b_l$; // postsmoothing
  }
}

Two-grid and Multigrid Methods II

- It is sufficient to set $\gamma = 1, \nu_1 = 1, \nu_2 = 0$ to get a iteration count of $\mathcal{O}(1)$.
- A single pass from level l to level 0 and back is denoted as V-cycle:



- Effort for two-dimensional structured grids: N is number of grid points in a row on the finest level, and $C := t_{\text{op}}$:

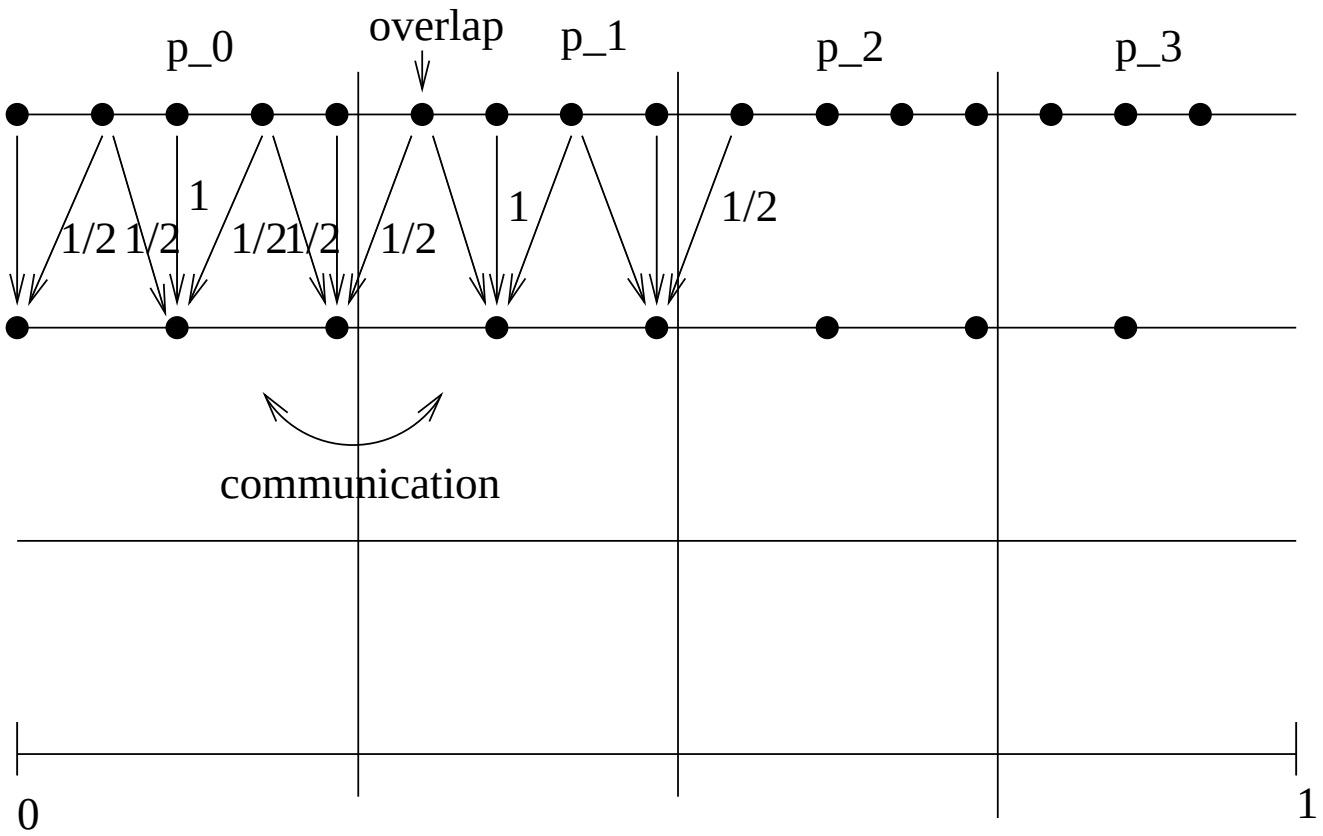
$$T_{\text{IT}}(N) = \underbrace{CN^2}_{\text{level l}} + \underbrace{\frac{CN^2}{4}}_{\text{level l-1}} + \frac{CN^2}{16} + \ldots + \underbrace{G(N_0)}_{\text{coarse grid}}$$

$$= CN^2 \underbrace{\left(1 + \frac{1}{4} + \frac{1}{16} + \ldots\right)}_{\frac{4}{3}} + G(N_0) = \frac{4}{3} CN_l + G(N_0)$$

Parallelisation I

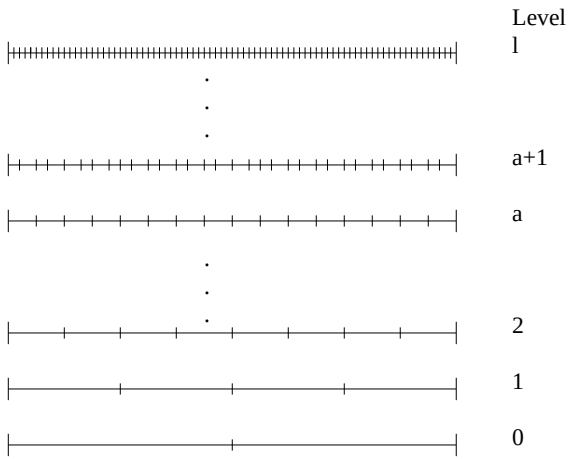
For data partitioning in the grid hierarchy on the individual processors one has to consider:

- In the coarse grid correction has to be checked, whether communication is necessary to calculate the node values in the coarse grid.
- How to handle the coarsest grids, where the number of unknowns in each dimension get smaller than the number of processors?



Parallelisation II

- For illustration of the method we only consider the one-dimensional case. The distribution in the high-dimensional case is according to the tensor product (chessboard-like).
- The processor limits are choosen at $p \cdot \frac{1}{P} + \epsilon$, such the nodes, that reside on the boundary between two processors, are still assigned to the „previous “.
- It is to remark, that the defect, that is restringated in the coarse grid correction, can only calculated on the single master node, but not in the overlap!
- To solve the problem with the decreasing node count in the coarsest grids, one uses successively fewer processors. Be for that $a := \mathrm{ld}\, P$ and again $C := t_{\mathrm{op}}$. On level 0 only one processor calculates, first on level a all are busy.



Parallelisation III

level	nodes	processors	effort
l	$N_l = 2^{l-a} N_a$	$P_l = P$	$T = 2^{l-a} C N_0$
a+1	$N_{a+1} = 2N_a$	$P_{a+1} = P$	$T = 2CN_0$
a	N_a	$P_a = P$	$T = CN_0$
2	$N_2 = 4N_0$	$P_2 = 4P_0$	$T = \frac{CN_2}{4} = CN_0$
1	$N_1 = 2N_0$	$P_1 = 2P_0$	$T = \frac{CN_1}{2} = \frac{C2N_0}{2} = CN_0$
0	N_0	$P_0 = 1$	$G(N_0) \overset{\text{be}}{\approx} CN_0$

- Let's consider the total effort: From level 0 to level a $\frac{N}{P}$ is constant, thus T_P grows like $\mathrm{ld}\, P$. Therefore we get

$$T_P = \mathrm{ld}\, P \cdot CN_0$$

- In the higher levels we get

$$T_P = C \cdot \frac{N_l}{P} \cdot \left(1 + \frac{1}{2} + \frac{1}{4} + \ldots\right) = 2C\frac{N_l}{P}$$

- The total effort is then given by the sum of both partial efforts.
- Here we have not taken into account the communication between the processors.

Parallelisation IV

How effects the usage of the multigrid method the number of iteration steps to be executed?

- We show the number of used processors against the choice of the grid spacing, that has been used.
- The minimal error reduction has been set to 10^{-6}.

P/l	5	6	7	8
1	10			
4	10	10		
16	11	12	12	
64	13	12	12	12

- The table shows the according iteration times in seconds for 2D (factor 4 of grid growths):

P/l	5	6	7	8
1	3.39			
4	0.95	3.56		
16	0.32	1.00	3.74	
64	0.15	0.34	1.03	3.85

Most Important Knowledge

- Jacobi scheme is one of the most simple iteration methods for the solution of linear equation systems.
- For fixed reduction factor ϵ one necessitates a specific number of iterations IT to reach a certain error reduction.
- IT is *independent* on the choice of the starting value, but depends directly from the choice of the method (e.g. Jacobi scheme) and the grid spacing h (thus N).
- For the Jacobi method applies $IT = O(h^{-2})$. For a halvening of the grid width h one needs the fourfold number of iterations to get the error reduction ϵ. Since an iteration costs also four times more, the effort has increased by a factor of 16!
- There are a series of better iteration schemes for which e.g. $IT = O(h^{-1})$, $IT = O(\log(h^{-1}))$ or even $IT = O(1)$ applies (CG method, hierarchical basis, multigrid method).
- Of course asymptotically ($h \to \infty$) each of these methods is superior to a parallel naive scheme.
- One should therefore at all parallelize the method with optimal sequential complexity, especially because we want to solve large problems (h small) on parallel machines.