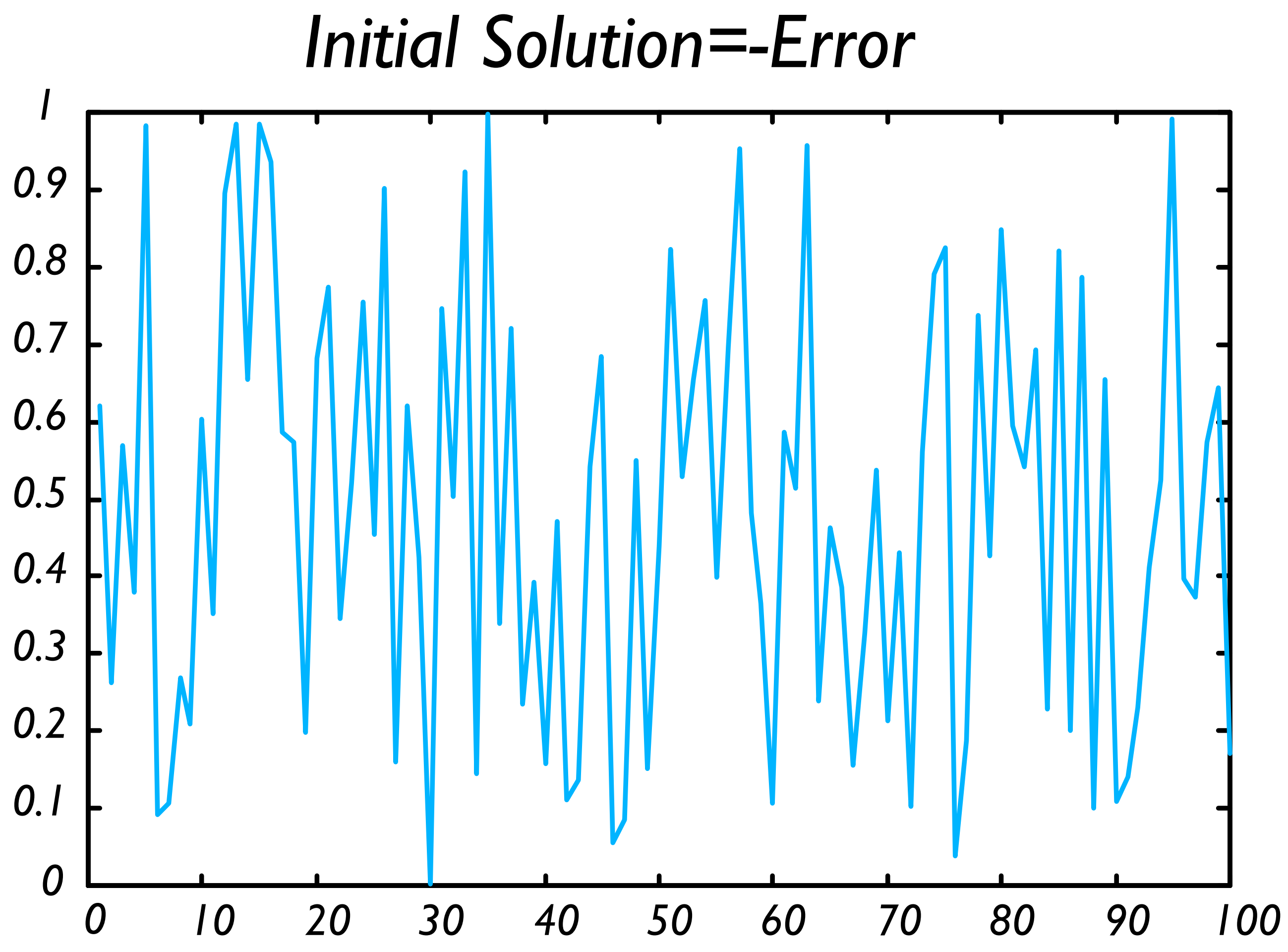
Iterative Methods and Multigrid

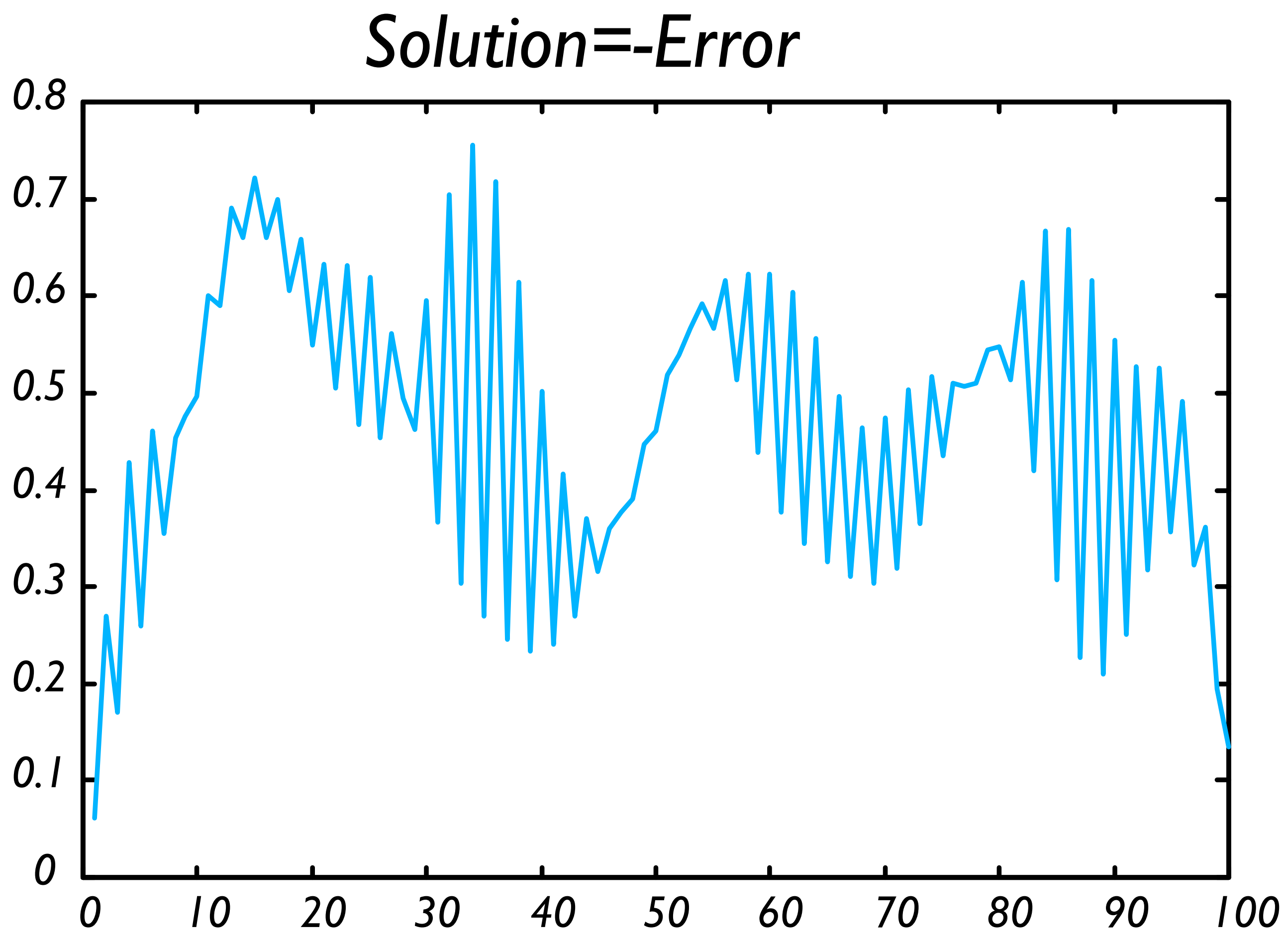
Part 1: Introduction to Multigrid

Error Smoothing



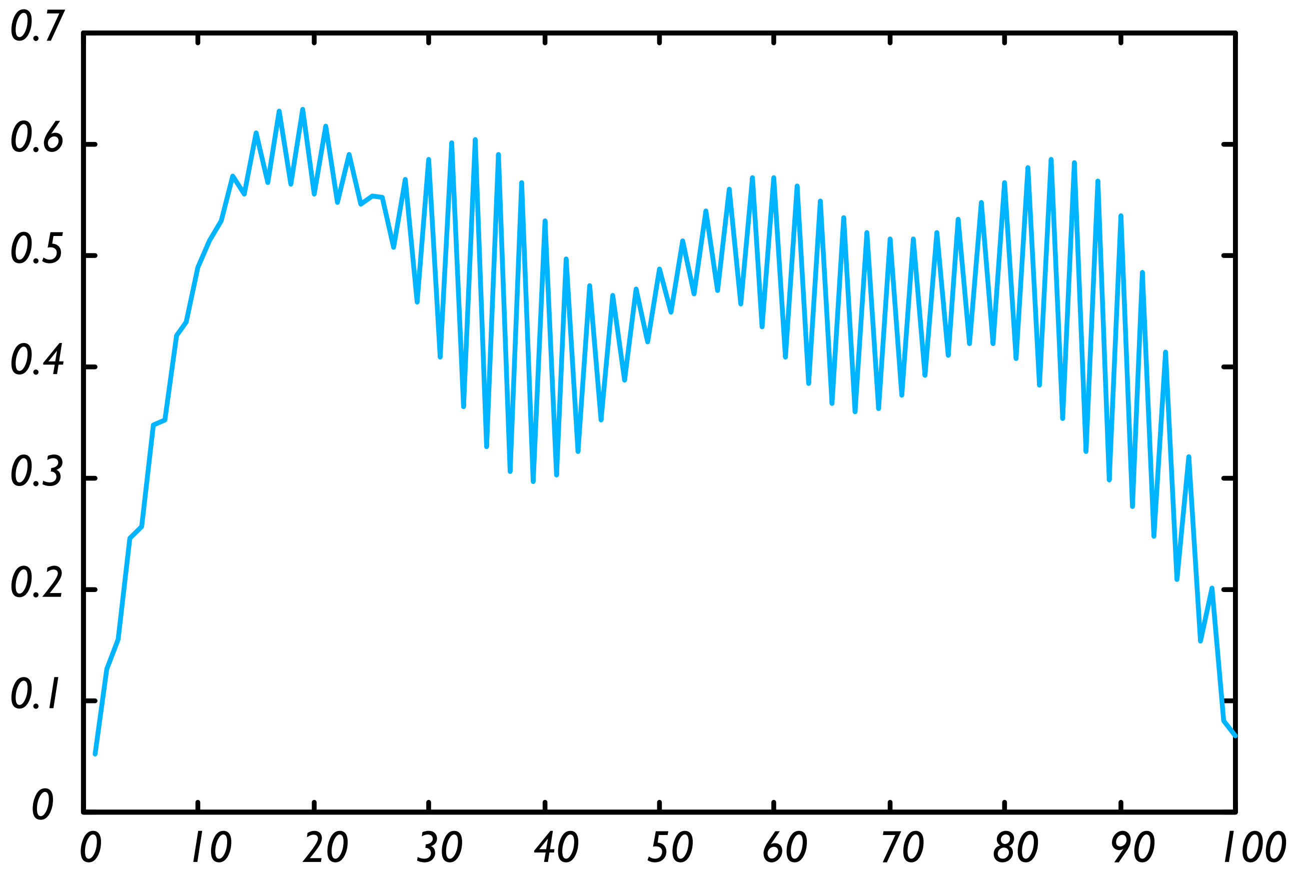
DCT: [4.9, 0.27, 0.16, 0.061, -2.4, ...], O(0.1) or O(0.2)

Error Smoothing

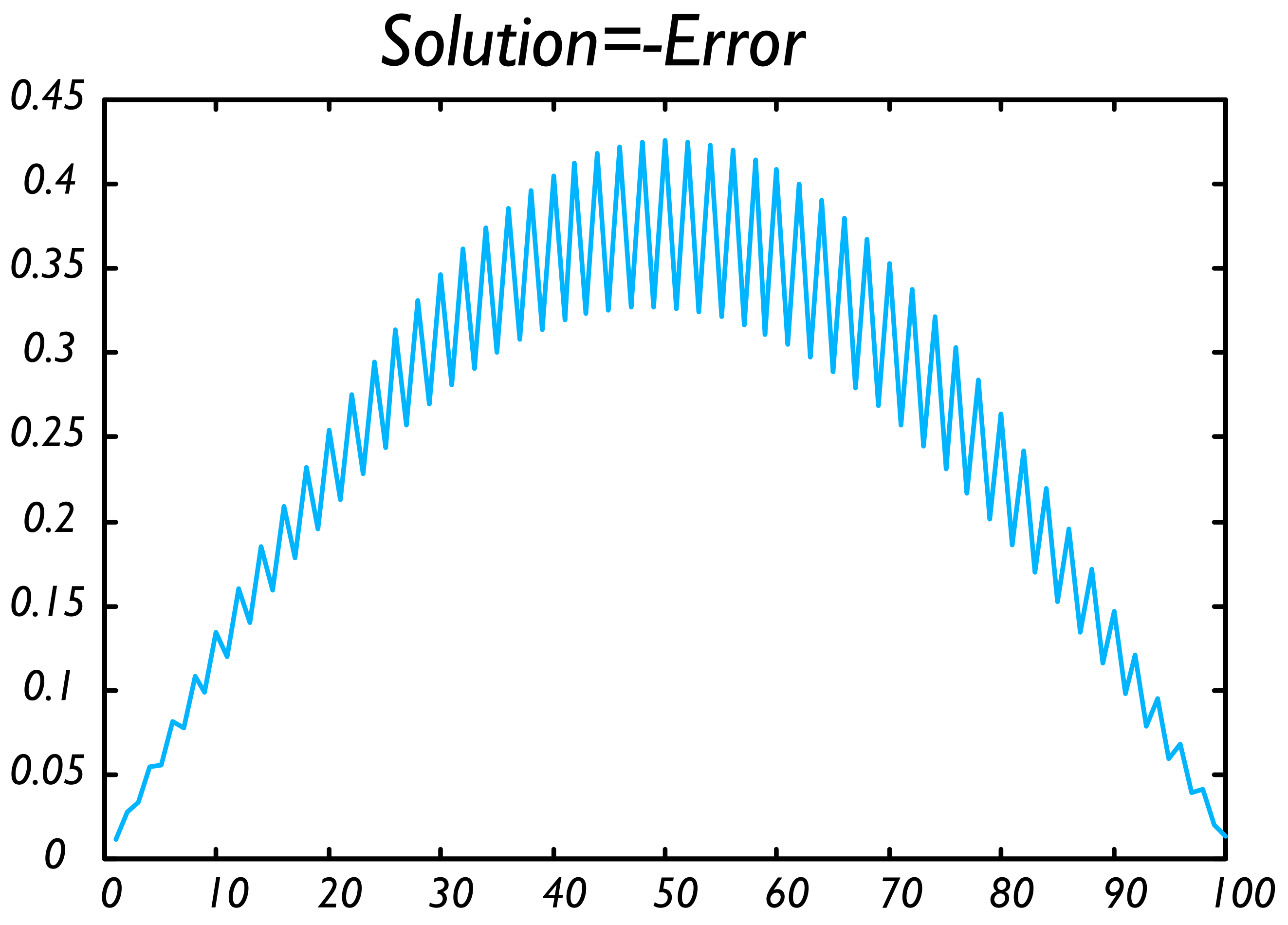


After 20 Jacobi iterations with $\omega=1$

Solution=-Error

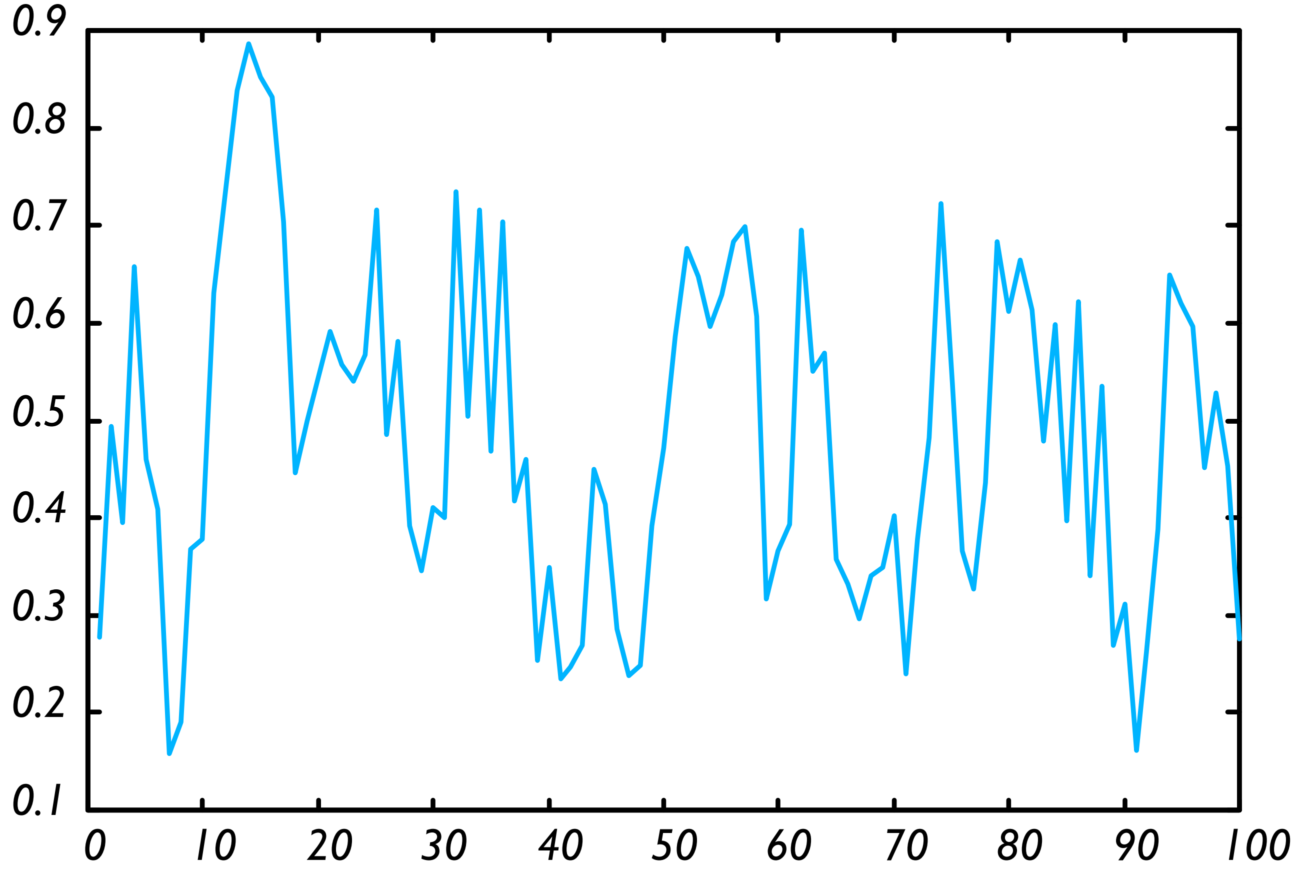


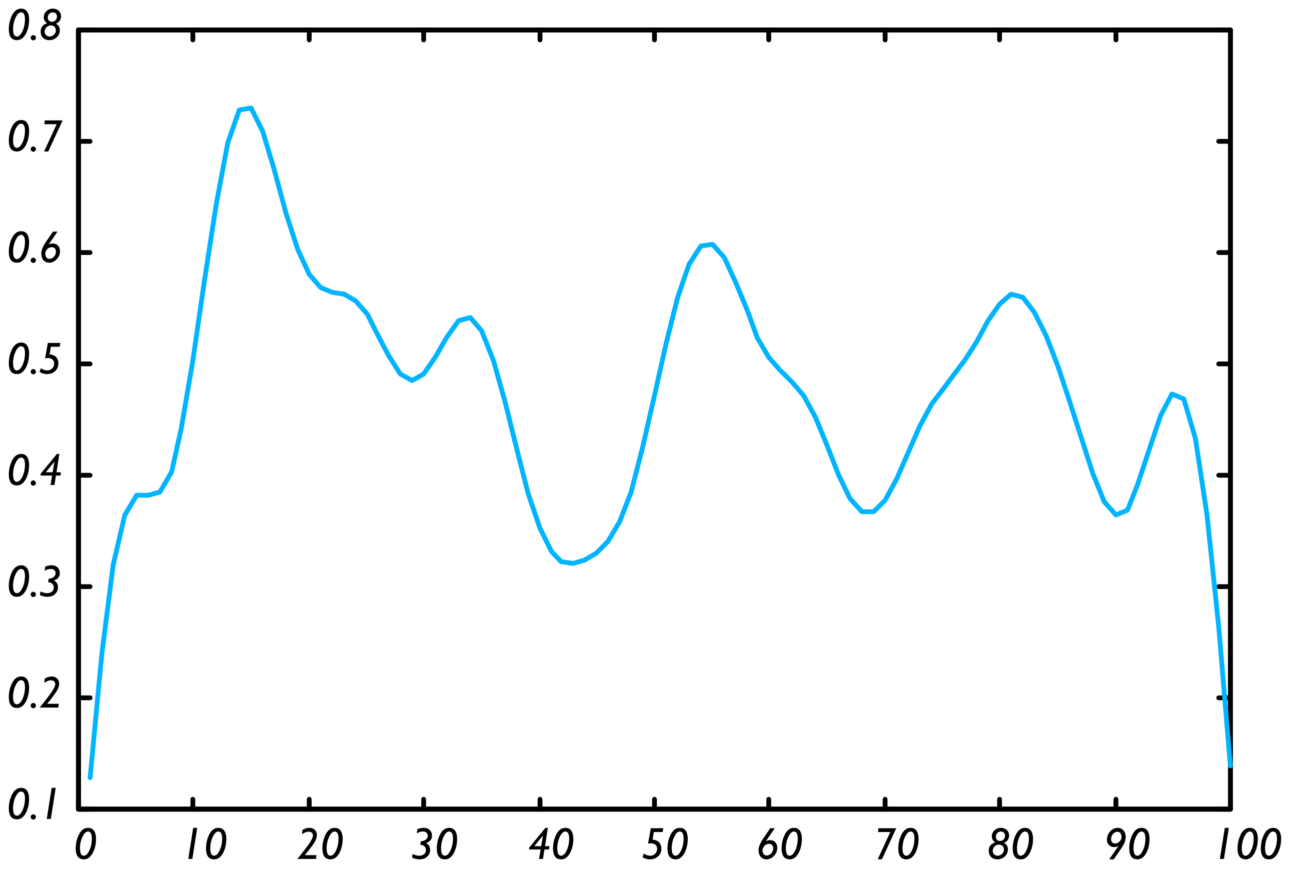
After 50 Jacobi iterations with ω=1

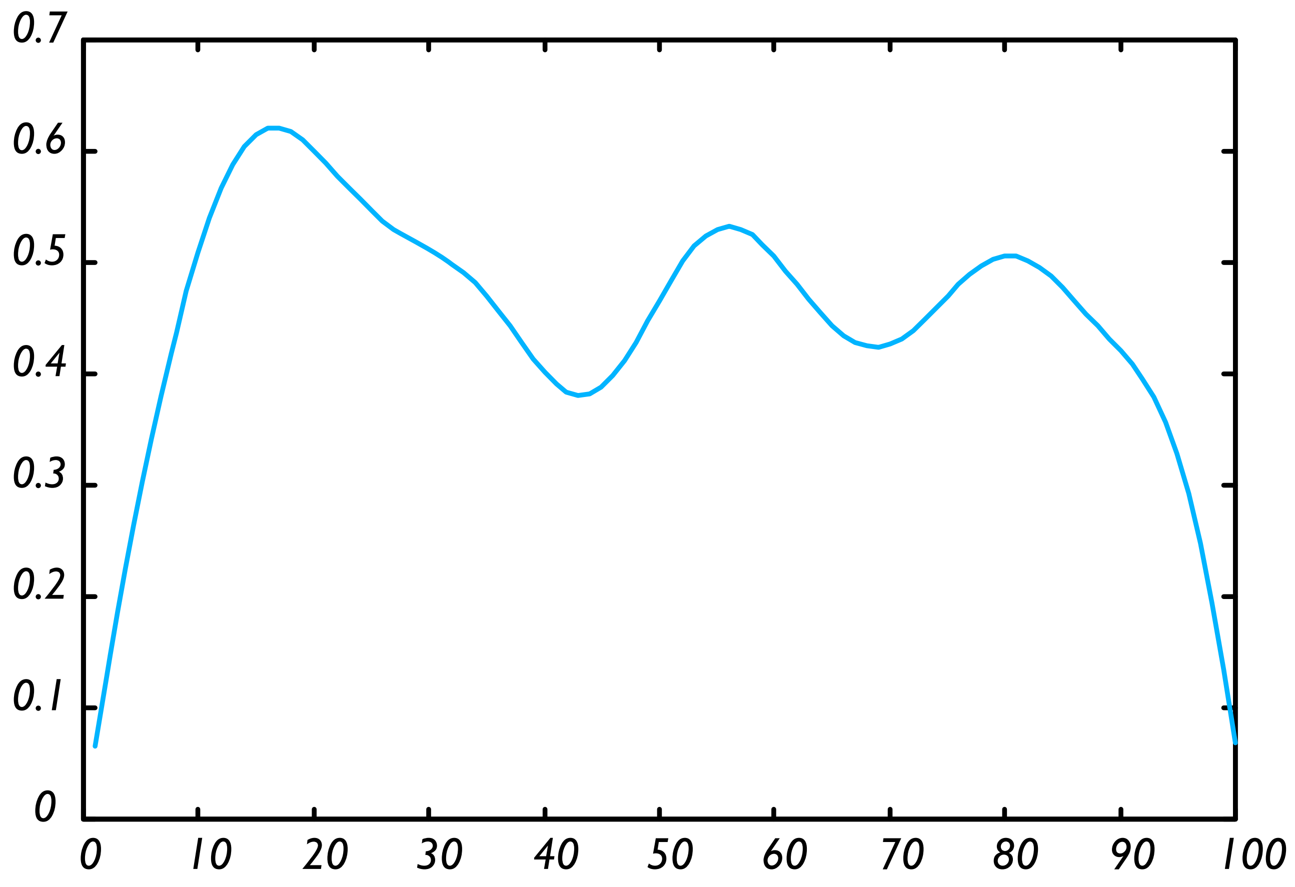


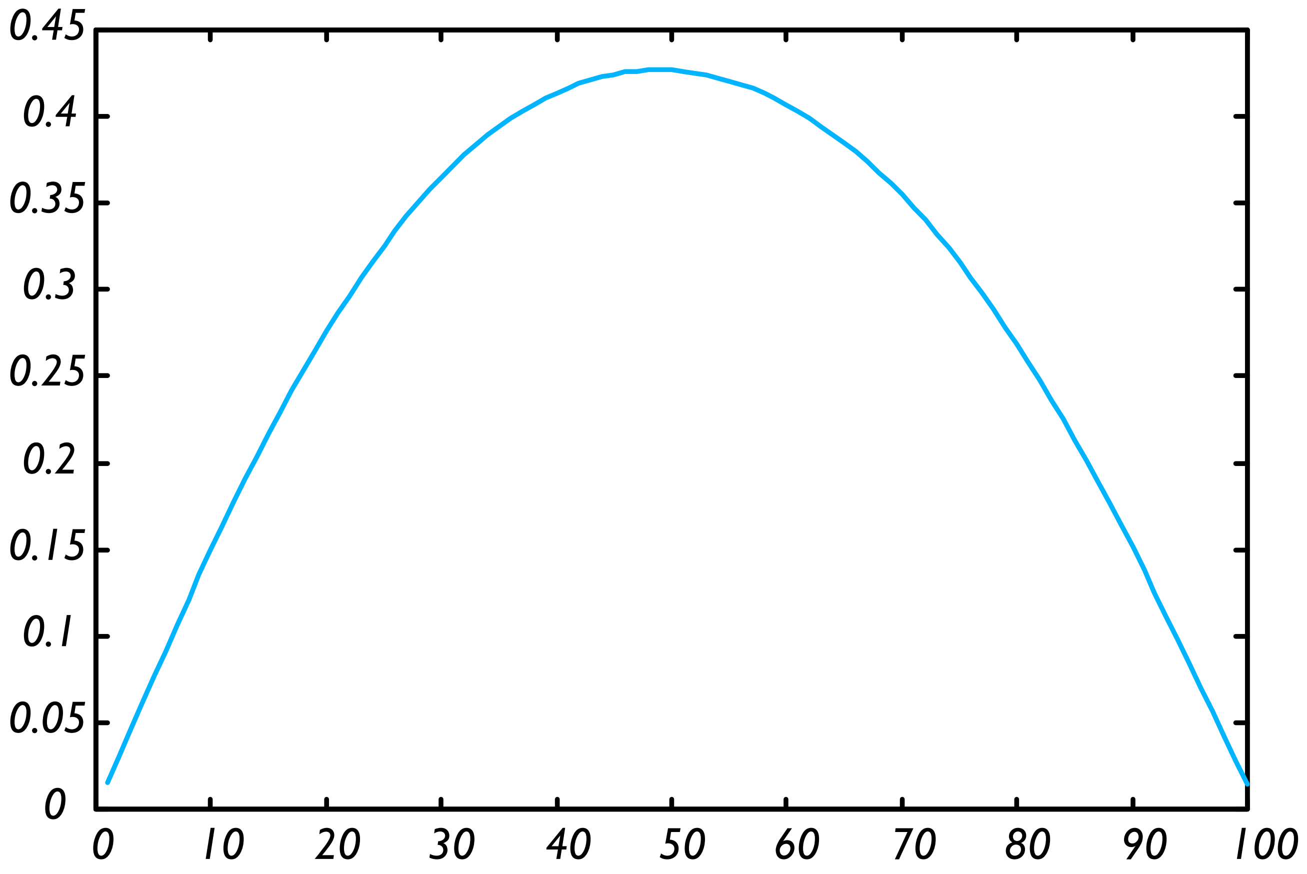
After 1000 Jacobi iterations with ω=1

DCT: [2.4, ε, -1.1, ε, ... , ε, 0.35], ε=O(0.01), mainly O(1e-3)


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10
20
30
40
50
60
70
80
90
100


0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10
20
30
40
50
60
70
80
90
100


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50
60
70
80
90
100


0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
10
20
30
40
50
60
70
80
90
100

Convergence: $e^{(k)} = G^k e^{(0)}$, where G is iteration matrix

$G = (I - P^{-1}A)$.

$u^{(k+1)} = u^{(k)} + \omega P^{-1} r^{(k)} = u^{(k)} + \omega P^{-1}(f - Au^{(k)}) =$

$(I - \omega P^{-1}A)u^{(k)} + \omega P^{-1} f$

Jacobi iteration matrix: $(1-\omega)I + \omega R_J = (1-\omega)I + \omega D^{-1}(D - A)$
This gives: $I - \frac{\omega}{2}A$

Eigenvalues $I - \frac{\omega}{2}A$: $\lambda(I - \frac{\omega}{2}A) = 1 - \frac{\omega}{2}\lambda(A)$

Eigenvalues of A

Eigenvalues of A:

Assume eigenvector close to physical eigenvector: $v_k = \sin \frac{\pi k j}{n}$

Apply pointwise rule for Jacobi:

$$Av_{j,k} = -\sin \frac{\pi k(j-1)}{n} + 2\sin \frac{\pi k j}{n} - \sin \frac{\pi k(j+1)}{n} = 2\sin \frac{\pi k j}{n} - 2\sin \frac{\pi k j}{n} \cos \frac{\pi k}{n} =$$

$2\sin \frac{\pi k j}{n}(1 - \cos \frac{\pi k}{n})$ (so eigenvector indeed)

$$1 - \cos \frac{\pi k}{n} = 1 - \cos \frac{2\pi k}{2n} = 1 - (1 - 2\sin^2(\frac{\pi k}{2n})) = 2\sin^2(\frac{\pi k}{2n})$$

$$2\sin \frac{\pi k j}{n}(1 - \cos \frac{\pi k}{n}) = \sin \frac{\pi k j}{n} \bullet 4\sin^2(\frac{\pi k}{2n})$$

This gives for the weighted Jacobi method: $I - \frac{\omega}{2}A$

$\lambda_{(R_{J,\omega})} = 1 - 2\omega \sin^2(\frac{\pi k}{2n})$ where $0 < \omega \leq 1$

Always converges. Poor convergence for which modes?

Some observations:
We see from experiments and analysis that convergence for smooth and oscillatory modes is very different. More precisely, our analysis (given some ω) how much each mode is reduced in norm per iteration (sweep over all points).

Choosing appropriate ω we can make all oscillatory modes converge relatively fast. However, no choice for ω exists that makes the convergence for modes with small k fast.

We could analyze this problem (Laplacian and Jacobi) easily because eigenvectors A are the eigenvectors of iteration matrix $(I - D^{-1}A)$. This is not generally the case.

The Jacobi iteration does not mix modes. The image of a sine wave under the iteration matrix is same wave damped. Not general.

Some terminology:

Consider vectors $v_k = \sin\frac{jk\pi}{n}$ on ‘normalized domain’: $[0,1]$
where $1 \leq k \leq n-1$ and $0 \leq j \leq n$

number of grid points: n
wavenumber: k
wavelength: $l = \frac{2}{k}$ (since n grid points span domain of size 1)
This also shows that mode k gives $\frac{k}{2}$ full sine waves on domain.

We cannot represent waves on our grid with a wavelength less than $2h$
This corresponds to wavenumber larger than n. Such waves would actually be disguised as waves with longer wavelength: aliasing.

The wavenumber $k = n/2$ corresponds to wavelength $l = 4/n \approx 4h$.

We saw that it is important to distinguish oscillatory and smooth waves:

Low frequency/smooth if $1 \le k < n/2$
High frequency/oscillatory if $n/2 \le k \le n-1$

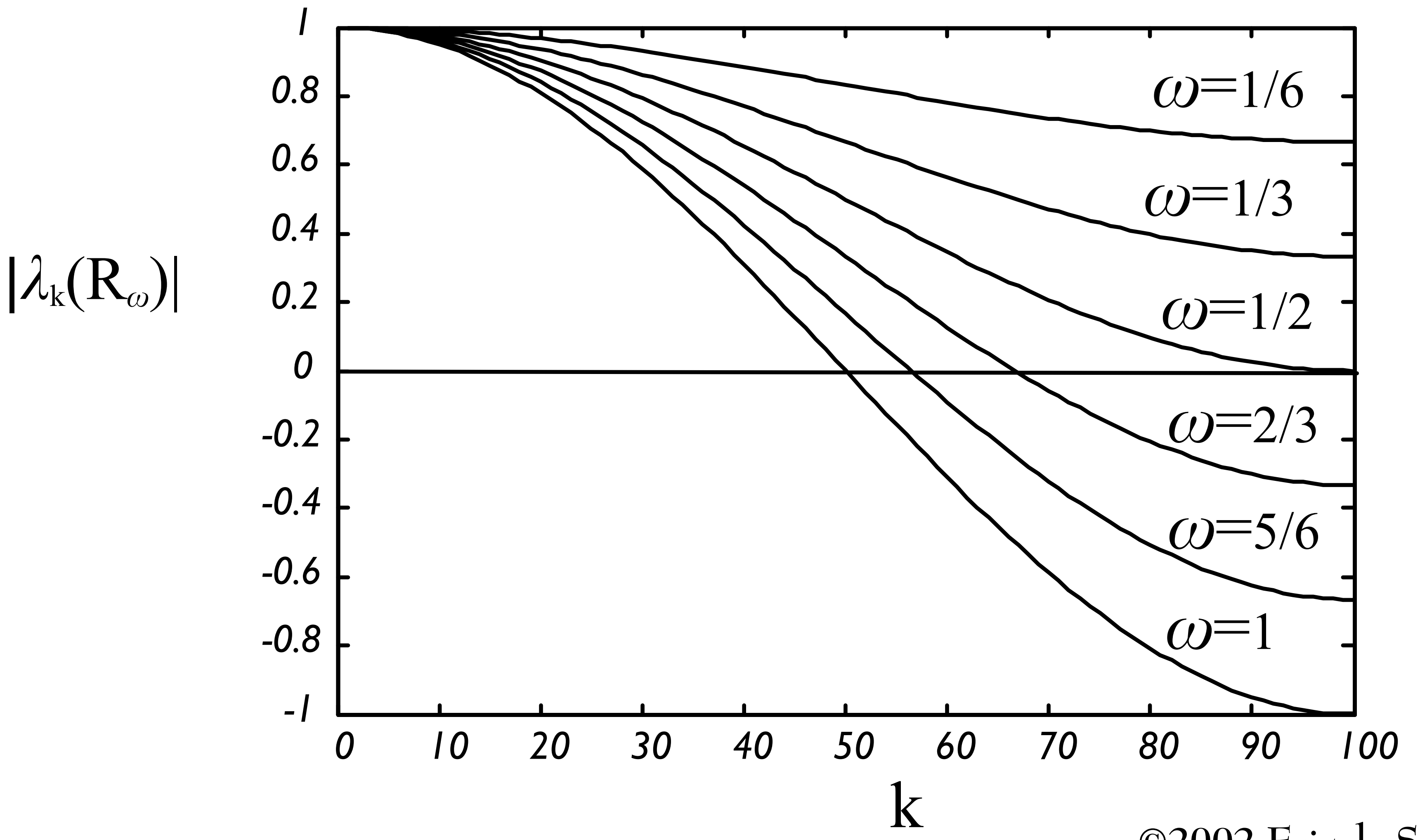
A particular iteration scheme (splitting) and omega give rise to the iteration
$u_{i+1} = (I - \omega P^{-1}A)u_i + \omega P^{-1} f$ which means for the error
$e_{i+1} = (I - \omega P^{-1}A)e_i = (I - \omega P^{-1}A)^i e_0$

So if $(I - \omega P^{-1}A) = V \Lambda V^{-1}$ and $e_0 = V\eta_0$ then $e_i = \Sigma_k v_k \lambda_k^i \eta_{k,0}$

So analogous to analysis of Jacobi for Laplacian we can simplify the analysis of the convergence by considering eigenvectors separately. We can consider which ω is best for which eigenvector (we can pick only one per iteration) and if there's a best ω overall.

Analyze which ω best for Laplacian and Jacobi iteration:

$\lambda_{(R_{J,\omega})} = 1 - 2\omega \sin^2\left(\frac{\pi k}{2n}\right)$ where $0 < \omega \leq 1$



©2002 Eric de Sturler

Standard Jacobi $\omega = 1$ works for the middle range wavenumbers, but does poorly for both low and high frequency waves.

The choice $\omega = 2/3$ does well for a fairly large range of wavenumbers

No ω does well for the lowest frequencies.

In fact $k = 1 : 1 - 2\omega \sin^2(\frac{\pi}{2n}) \approx 1 - 2\omega \sin^2(\frac{h\pi}{2}) \approx 1 - \frac{\omega h^2 \pi^2}{2}$

So for small h there will be no $\omega \le 1$ that will make $|\lambda_1(R_\omega)|$ small

Worse, for as $h \to 0$ (solving problem more accurately) the reduction factor becomes closer and closer to 1.

Let's give up on the low frequencies and focus on high frequencies.
(or at least postpone elimination of low frequency error)

oscillatory modes: $n/2 \le k \le n-1$
We see that several values for ω do well. To get 'the best' we require again the equi-oscillating one (remember Chebyshev polynomials):

$$\lambda_{n/2}(R_\omega) = -\lambda_n(R_\omega)$$

This gives

$$\begin{aligned}
&1 - 2\omega \sin^2\left(\tfrac{(n/2)\pi}{2n}\right) = -1 + 2\omega \sin^2\left(\tfrac{n\pi}{2n}\right) \Leftrightarrow \\
&1 - 2\omega \sin^2\left(\tfrac{\pi}{4}\right) = -1 + 2\omega \sin^2\left(\tfrac{\pi}{2}\right) \Leftrightarrow \\
&1 - \omega = -1 + 2\omega \Leftrightarrow \\
&2 = 3\omega \Leftrightarrow \\
&\omega = \tfrac{2}{3}
\end{aligned}$$

Worst convergence factor attained at $k = \frac{2}{3} : 1 - \frac{4}{3}\sin^2\left(\frac{\pi}{4}\right) = 1 - \frac{4}{6} = \frac{1}{3}$.

So $|\lambda_k| \leq 1/3$ and oscillatory components are reduced by at least a factor 3 each iteration.

This factor is called the smoothing factor.
From its derivation we see that it is independent of h.

Suppose we wanted to reduce smooth modes by at least factor $1/2$.

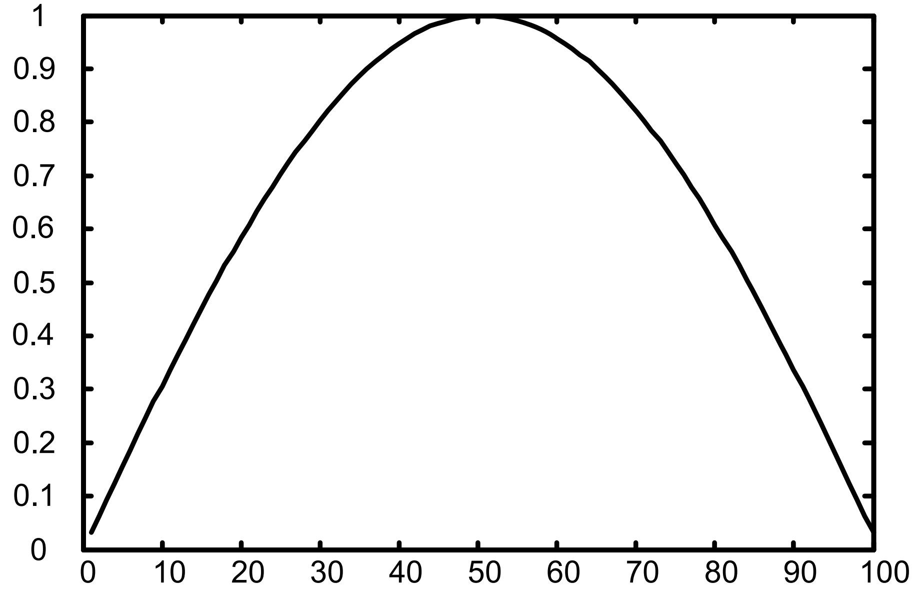
$$1 - 2\omega \sin^2\left(\tfrac{\pi h}{2}\right) \approx 1 - 2\omega \tfrac{\pi^2 h^2}{4} = 1 - \tfrac{\omega \pi^2 h^2}{2} \rightarrow \omega \pi^2 h^2 = 1 \Rightarrow \omega = \tfrac{1}{\pi^2 h^2}$$

Then for $k = n/2$ we get:

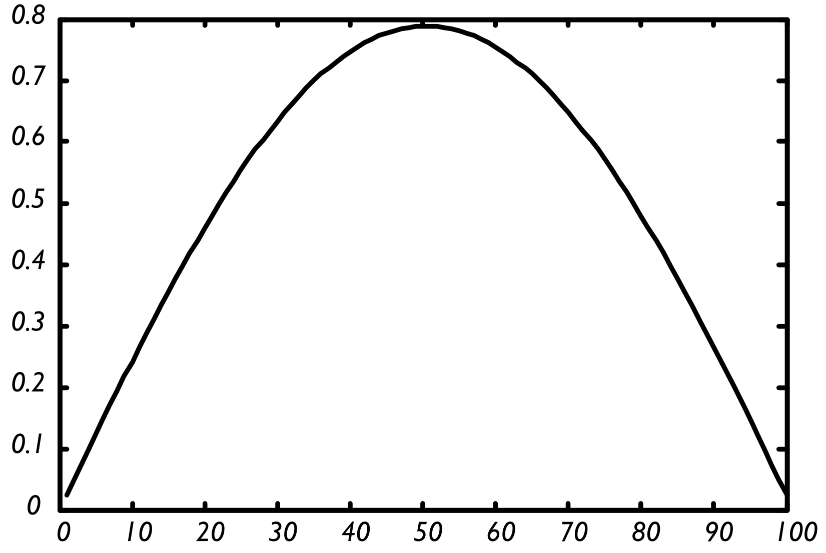
$$1 - \tfrac{2}{\pi^2 h^2} \sin^2\left(\tfrac{\pi}{4}\right) = 1 - \tfrac{1}{\pi^2 h^2}$$

So for $h < \frac{1}{\pi\sqrt{2}}$ we will amplify the oscillating modes! Divergence.

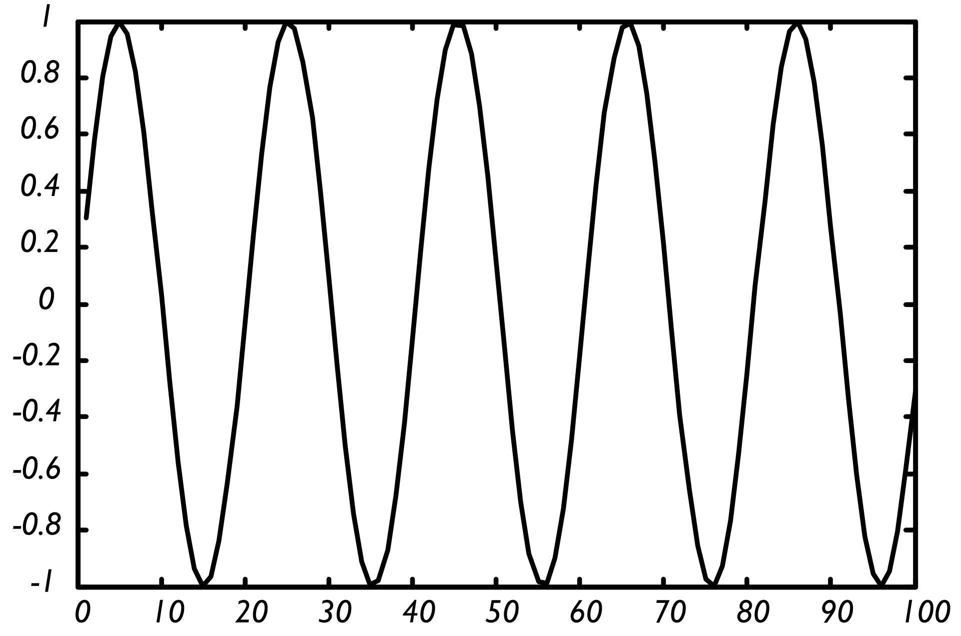
k=1



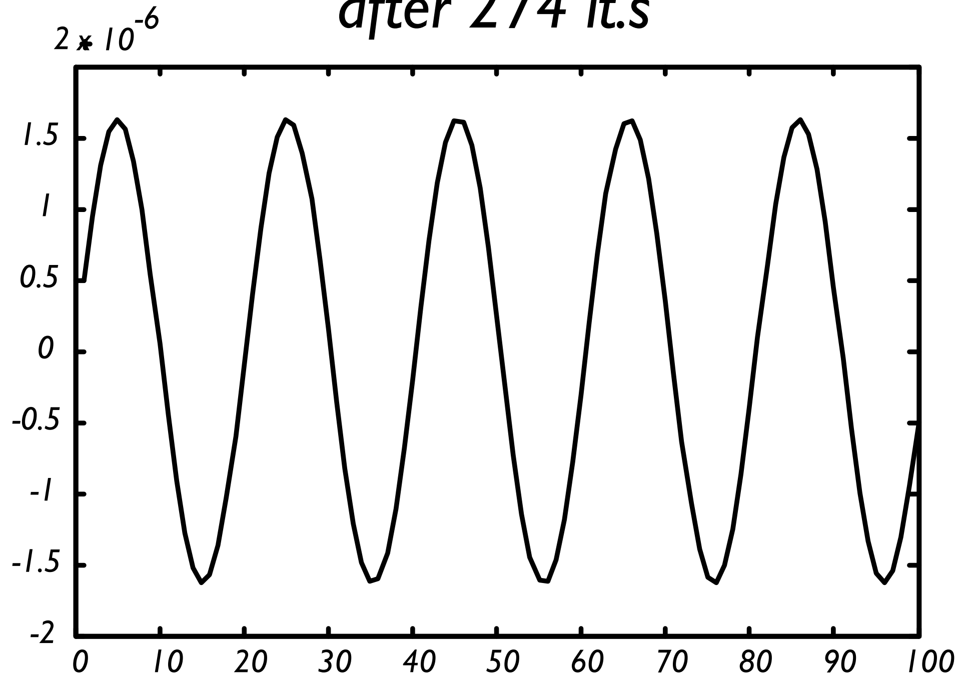
after 500 iterations



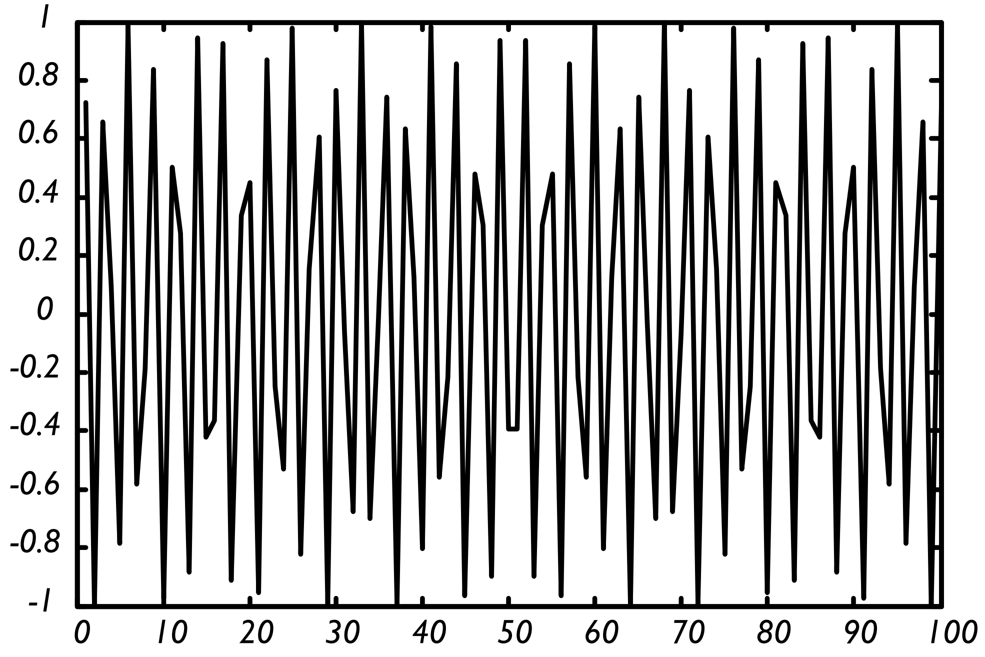
$k=10$



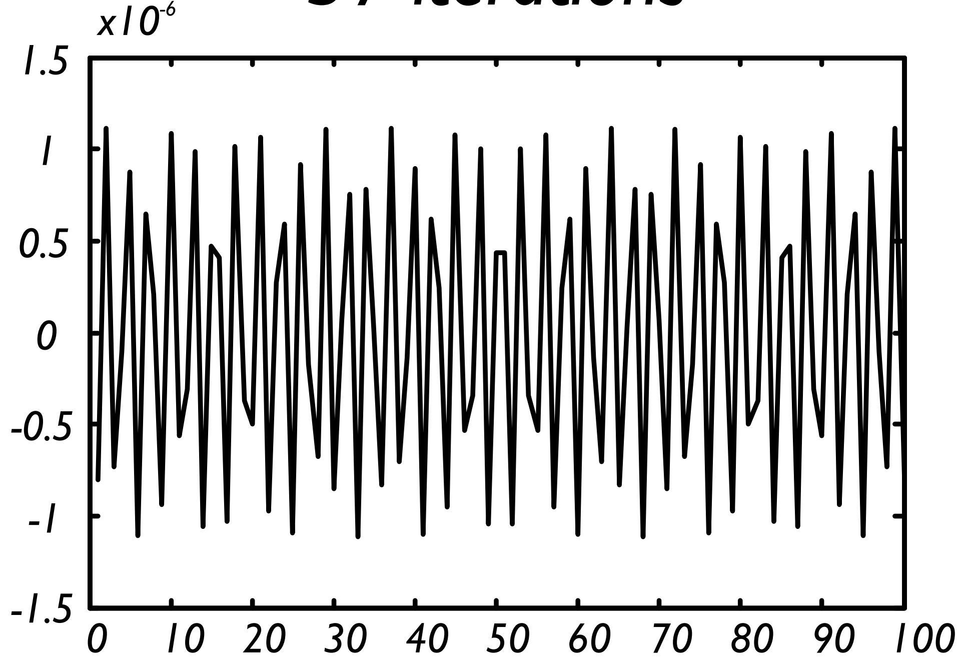
after 274 it.s



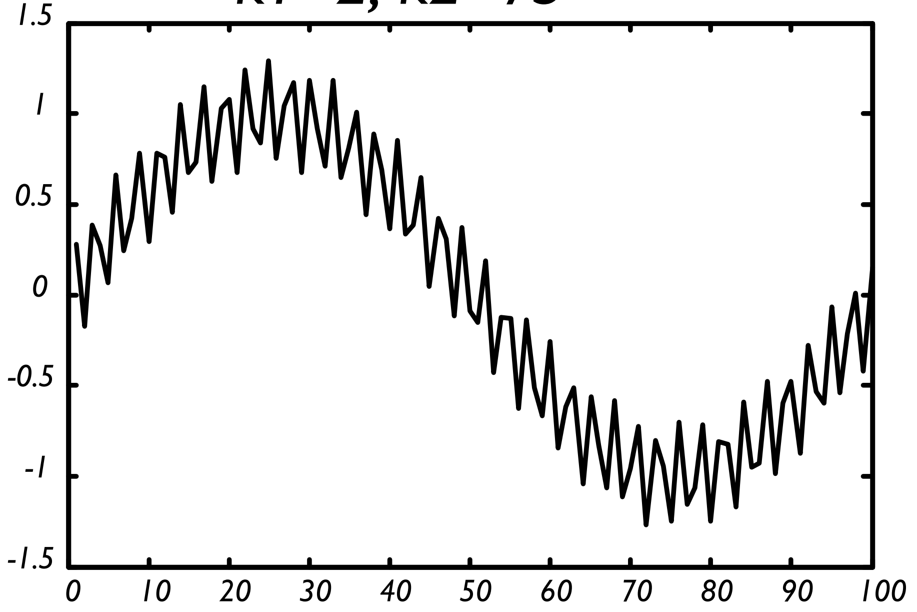
k=75



37 iterations

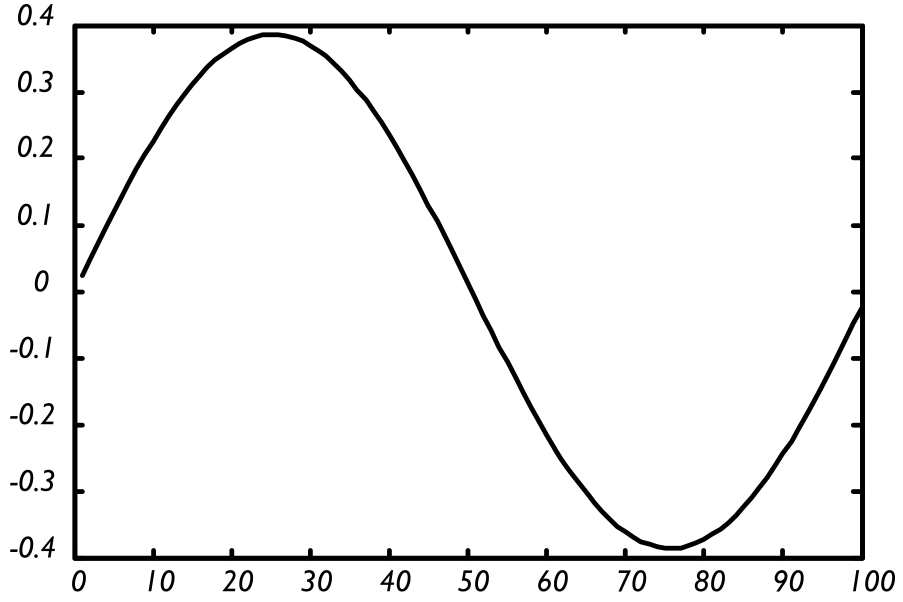


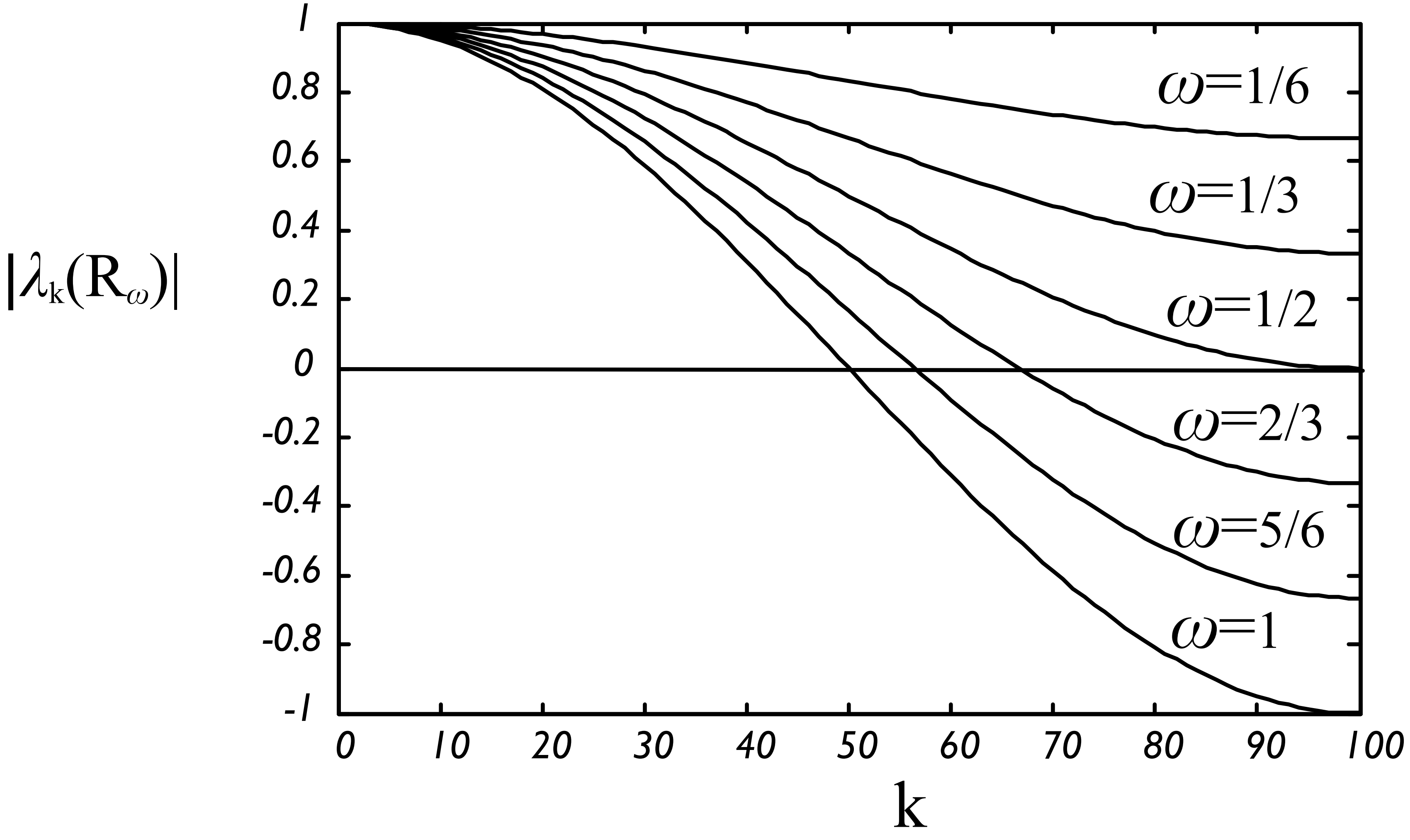
k1=2; k2=75



after 500 iterations

coarse part removed in a few iterations





$|\lambda_k(R_\omega)|$
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
10
20
30
40
50
60
70
80
90
100
k
ω=1/6
ω=1/3
ω=1/2
ω=2/3
ω=5/6
ω=1

Waves on different grids

Consider wave with wavenumber k:

$u^h_{j,k} = \sin\frac{jk\pi}{n}$ on grid Ω^h

Consider same wave on the coarse grid (half the points), by simply taking value at every other point:

$u^{2h}_{j,k} = u^h_{2j,k} = \sin\frac{2jk\pi}{n} = \sin\frac{jk\pi}{n/2}$ on grid Ω^{2h}.

So kth mode on fine grid gives kth mode on coarse grid.

Should not be surprising since kth mode is wave with wavelenght $\frac{2}{k}$, which does not depend on n.
Another way to see this is that we have half the number of points and half the number of waves.

Waves on different grids

The oscillatory waves on Ω^{2h}, which has $n/2$ points, are waves with wavenumbers $n/4 \leq k < n/2 - 1$.

Since wave number does not change, the modes with wavenumber $n/4 \leq k < n/2$ become oscillatory. So half the smooth modes on Ω^{h} become oscillatory on Ω^{2h}. The other half remain smooth (but less smooth than before).

The oscillatory modes on the fine grid cannot be represented on the coarse grid.

What happens to the oscillatory modes on Ω^{h} when we go to Ω^{2h}?

Why is this a problem for our solution algorithm?

Waves on different grids

Given problem with $n = 32$ with gives 33 grid points and $j = 0 \ldots 32$.

We look at following waves on $\Omega^{1/32}$ and $\Omega^{1/16}$:
1) $k = 5, 14, 26$
2) $k = 5, 14, 30$

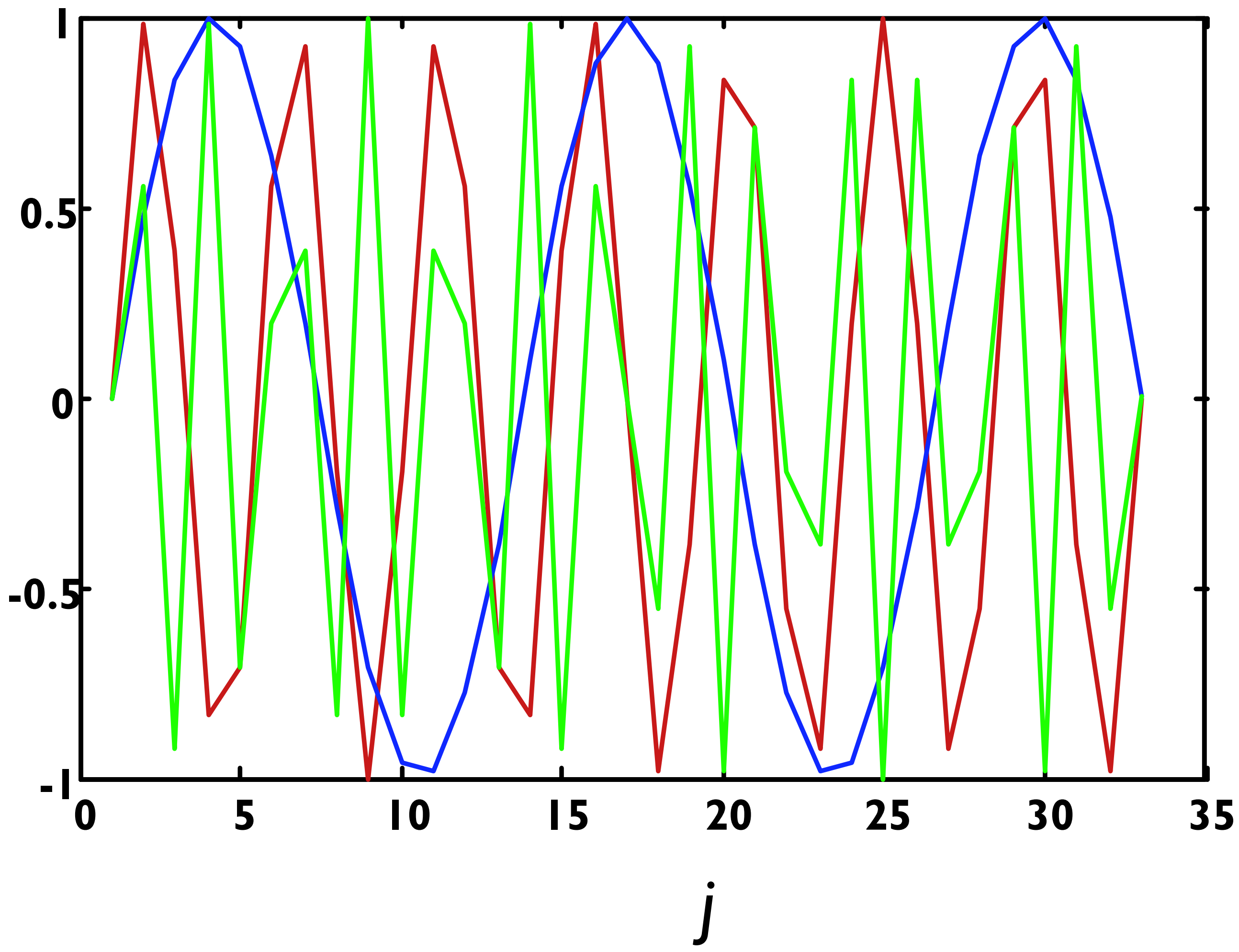
Notice that
$k = 5$ is smooth on both grids (but less smooth on $\Omega^{1/16}$)
$k = 14$ is smooth on $\Omega^{1/32}$ and becomes oscillatory on $\Omega^{1/16}$
$k = 26, 30$ are oscillatory on $\Omega^{1/32}$ and become smooth on $\Omega^{1/16}$

The effect for $k = 26, 30$ is called aliasing: an oscillatory wave disguises itself as a smooth one. In fact a wave with wavenumber $k > n-1$ appears as a wave with wavenumber $k = 2n - k$.

There is no way to distinguish the two on the given grid.

Waves on different grids

$$\sin \frac{jk\pi}{n}$$



$k1=5$

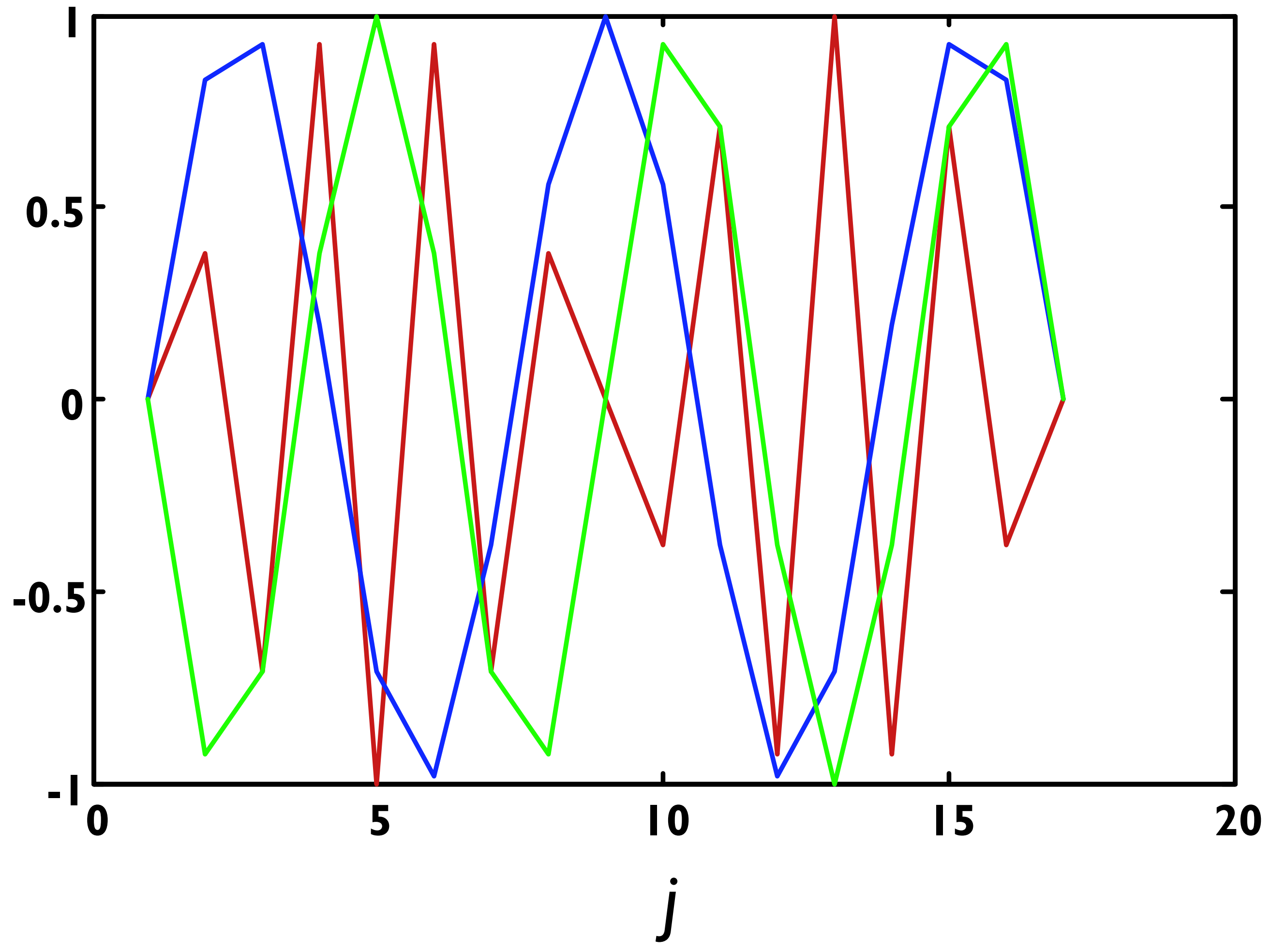
$k2=14$

$k3=26$

$n=32$

Waves on different grids

$$\sin \frac{jk\pi}{n/2}$$



$k1=5$

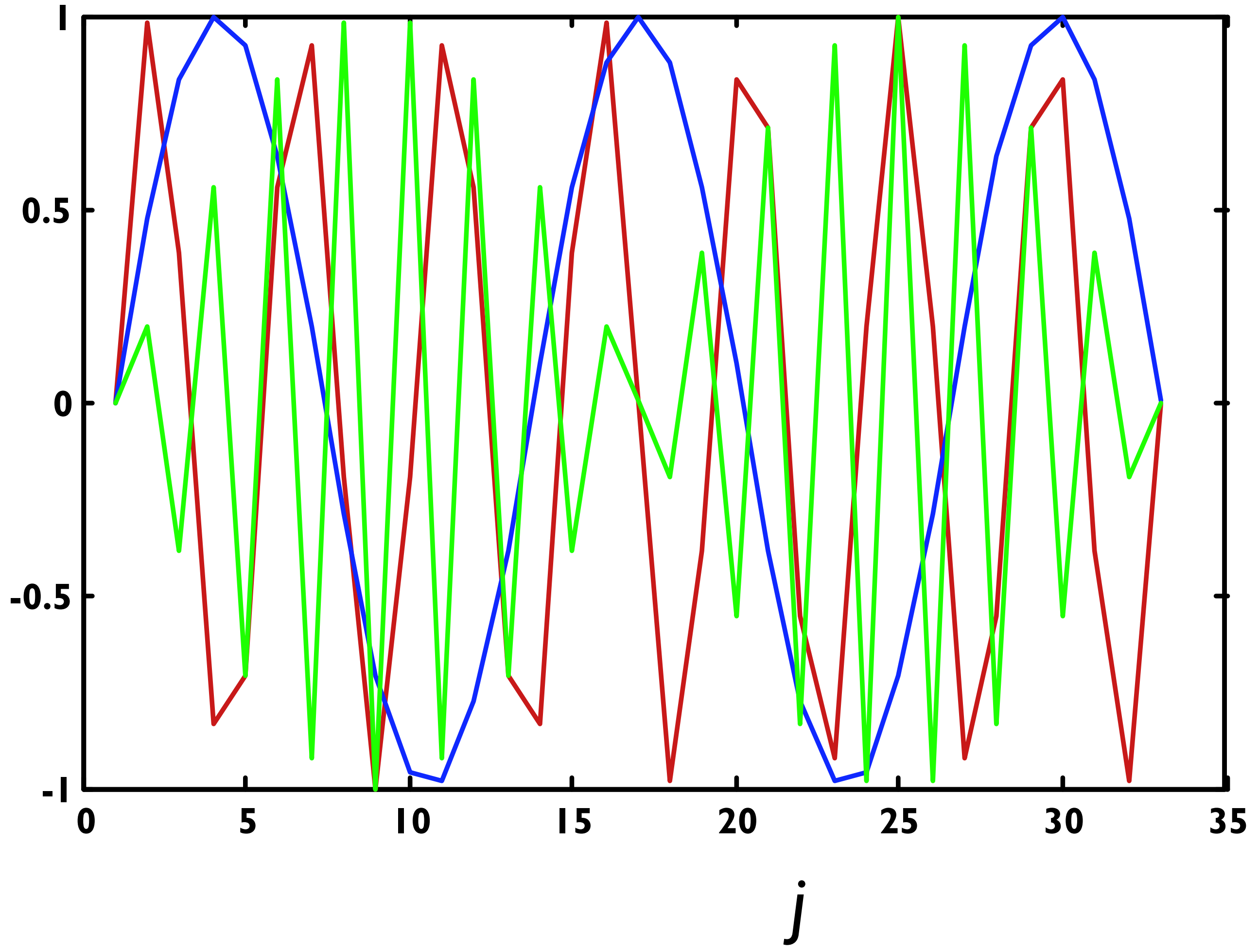
$k2=14$

$k3=26$

$n=16$

Waves on different grids

$\sin \frac{jk\pi}{n}$



$k1=5$

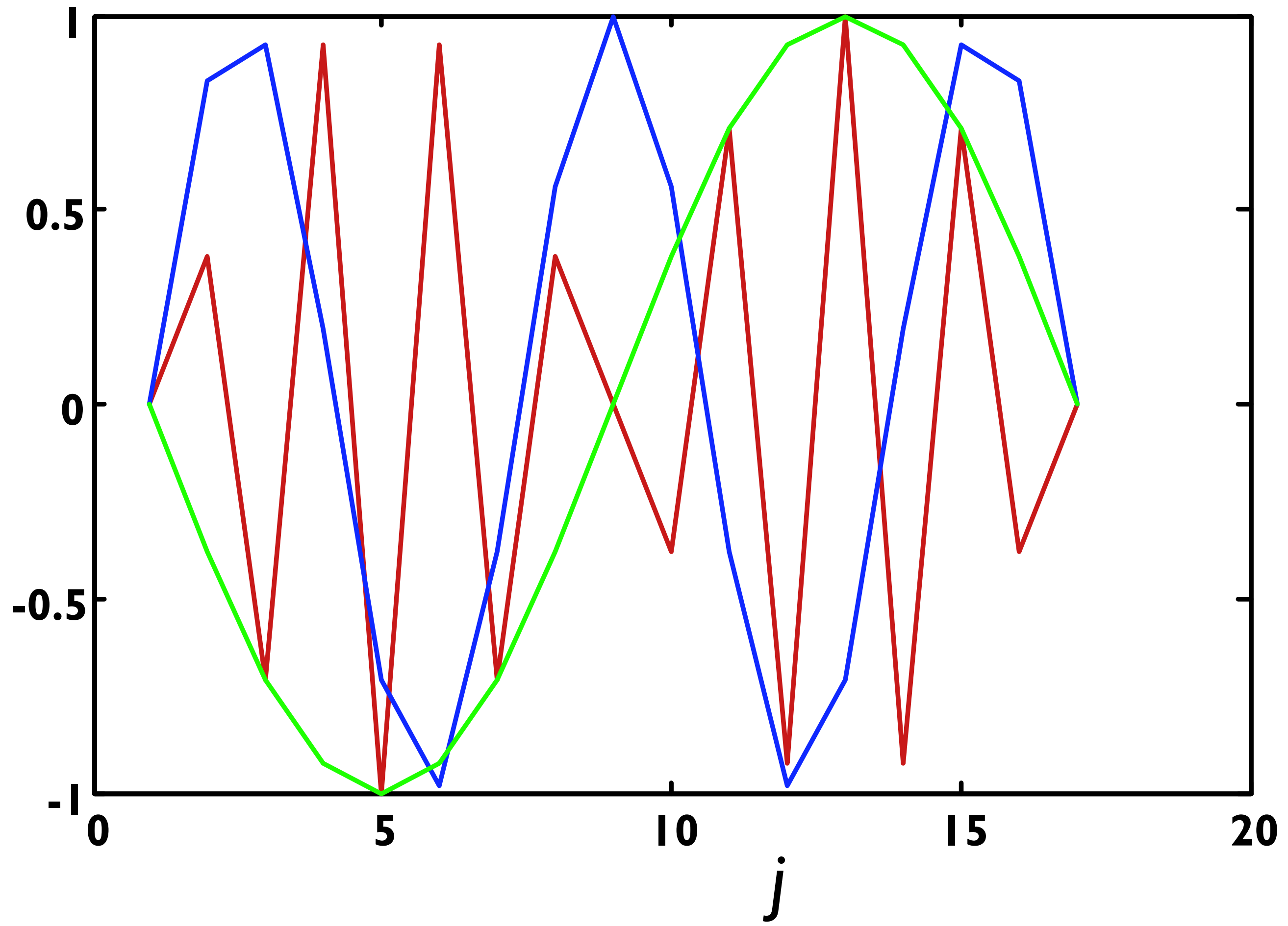
$k2=14$

$k3=30$

$n=32$

Waves on different grids

$$\sin \frac{jk\pi}{n/2}$$



k1=5

k2=14

k3=30

n=16

The basic idea behind MG

We saw that our basic iteration or relaxation methods are good in damping the oscillatory components of the error. They are not effective on the smooth modes.

So the idea is to use relaxation on the fine grid until oscillatory modes are sufficiently reduced and convergence slows down.

Then we move to the coarse grid where the (fine grid) smooth modes are more oscillatory and hence more effectively reduced by relaxation.

We saw that the very smooth modes ($k < n/4$) on the fine grid remain smooth on the coarse grid. Hence they will still be reduced slowly by relaxation.

The natural answer to this is to move to the next coarser grid, etc.

Typically at some point a direct solver is used.

Grid transfer and coarse grid eq.s

Important question now is how do we move among the grids.

1) How do we use the solution from the coarser grids (that have reduced or removed the smooth components of error) to get a better solution on the fine grid?

2) How do we transfer the approximate solution from fine grid to coarse grid and how do we derive the equations to solve on the coarse grid?

Grid transfer and coarse grid eq.s

Let's derive equations for the coarse grid.

We want to compute a correction to the solution on the fine grid. That means we must compute an approximation to the error.

The residual gives us an equation to compute the error: $Ae = r$

So we compute the residual at Ω^h and map it to Ω^{2h}. This can be done in many ways, but for the moment we stick to the simple trivial injection we saw before: $r_j^{2h} = r_{2j}^h$. This will provide our right hand side.

Again to keep things simple we assume that the coarse grid operator is the coarse grid discretization of the PDE (basically same operator as on the fine grid): A^{2h} (fine grid A^h).

So we have the equation: $A^{2h}e^{2h} = r^{2h}$

Grid transfer and coarse grid eq.s

Suppose we find a satisfactory approximation to the error e^{2h}. How do we use this to correct the fine grid approximation to the solution?

This operation is called interpolation or prolongation. Again many possibilities exist. We will look at a simple one first. Clearly for the points existing in both grids we can simply take the same values (just as going from fine to coarse grid). For the intermediate points we use linear interpolation. An important rationale for this is the following.

Assume the error on the fine grid is smooth. Further assume that we have the exact coarse grid error on the coarse grid. Linear interpolation of the coarse grid error on the fine grid will give a smooth function. Hence we may assume we get a good approximation to the fine grid error.

On the other hand if the fine grid error is oscillatory we cannot hope to get a good approximation (hence e^h should be smooth).

Grid transfer and coarse grid eq.s

The interpolation or prolongation operator is represented by I_{2h}^{h}.
We will represent the restriction operator by I_{h}^{2h}.

Assume n is even, $(n+1)$ grid points, and we ignore the boundary points and assume zero (Dirichlet) boundary conditions. Then we need to map from coarse grid points $1\ldots\frac{n}{2}-1$ to fine grid points $1\ldots n-1$:

$$I_{2h}^{h}:\mathbb{R}^{\frac{n}{2}-1}\rightarrow\mathbb{R}^{n-1}=\frac{1}{2}\begin{pmatrix} 1 & & \\ 2 & & \\ 1 & 1 & \\ & 2 & \\ & 1 & \\ & & \ddots \end{pmatrix}$$

The operator takes the coarse grid value at even grid points and averages values at neighboring points for the odd points. If we include the points on the boundary it would copy the values (even points).

Grid transfer and coarse grid eq.s

We will discuss two restriction operators: I_h^{2h}
Injection simply assigns the fine grid values to the corresponding coarse grid variables:

$$I_h^{2h} : \mathbb{R}^{n-1} \to \mathbb{R}^{\frac{n}{2}-1} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 1 & \\ & & \vdots & & & \ddots \end{pmatrix}$$

Another restriction operator is so-called full-weighting.
Here the values at the coarse grid points are taken as weighted averages of their (immediate) neighbors. The full weighting operator is a scaled transpose of the linear interpolation (prolongation) operator.

(compare with the domain decomposition operators)

Grid transfer and coarse grid eq.s

Full weighting at gives for a coarse grid variable:

$$v_j^{2h} = \frac{1}{4}\left(v_{2j-1}^h + 2v_{2j}^h + v_{2j+1}^h\right)$$

Written as a matrix operator (for whole grid) this gives

$$I_h^{2h} : \mathbb{R}^{n-1} \to \mathbb{R}^{\frac{n}{2}-1} = \frac{1}{4}\begin{pmatrix} 1 & 2 & 1 & & & \\ & & 1 & 2 & 1 & \\ & & & & & \ddots \end{pmatrix}$$

Note that all these steps are very easy to implement as local operators at each point.

2-Grid algorithm

We now have all the steps to put our first multigrid algorithms together.

Two grid algorithm: $Au = f$ (on Ω^h)

Ω^h: Do k_1^h relaxations on $A^h u^h = f^h$ starting with some initial guess

$r^h = f^h - A^h u^h; \quad r^{2h} = I_h^{2h} r^h;$

Ω^{2h}: Solve (directly) $A^{2h} e^{2h} = r^{2h}$

$\hat{e}^h = I_{2h}^h e^{2h}; \quad u^h = u^h + \hat{e}^h;$

Ω^h: Do k_2^h relaxations on $A^h u^h = f^h$ starting with new u^h.

A direct solver is used for $A^{2h} e^{2h} = r^{2h}$. Obviously for very large systems this is not practical. However, the two grid algorithm is often used for analysis.

An obvious alternative for a direct solver is to do another coarse grid correction on Ω^{4h} and so on (CS students feel the recursion coming).

2-Grid algorithm

We may also use an iterative solver (another couple of relaxation sweeps) on Ω^{2h}: $A^{2h}e^{2h} = r^{2h}$. This can in fact already be surprisingly effective.

We typically use a direct solver at some level (when the number of variables gets sufficiently small).

V-cycle variations

V-cycle MG algorithm: $Au = f$(on Ω^h)

Ω^h: Do k_1^h relaxations on $A^h u^h = f^h$ starting with some initial guess

$r^h = f^h - A^h u^h; \quad f^{2h} = I_h^{2h} r^h;$

Ω^{2h}: Do k_1^{2h} relaxations on $A^{2h} u^{2h} = f^{2h}$ starting with zero guess

$r^{2h} = f^{2h} - A^{2h} u^{2h}; \quad f^{4h} = I_{2h}^{4h} r^{2h};$

and so on for coarser and coarser grids

Ω^{Lh}: Solve (directly) $A^{Lh} u^{Lh} = f^{Lh}$; (suff. iterations make direct solve)

$e^{(L-1)h} = I_{Lh}^{(L-1)h} u^{Lh};$

$\Omega^{(L-1)h}$: $u^{(L-1)h} = u^{(L-1)h} + e^{(L-1)h};$

Do $k_2^{(L-1)h}$ relaxations on $A^{(L-1)h} u^{(L-1)h} = f^{(L-1)h}$ using new $u^{(L-1)h}$

and so on for finer and finer grids

Ω^h: $u^h = u^h + e^h;$

Do k_2^h relaxations on $A^h u^h = f^h$ starting with new u^h.

V-cycle variations

Let’s experiment: