POISSON'S EQUATION - DISCRETIZATION

The Dirichlet boundary value problem for Poisson's equation is given by

$$\begin{aligned} \Delta u(x,y) &= g(x,y), \quad (x,y) \in R \\ u(x,y) &= f(x,y), \quad (x,y) \in \Gamma \end{aligned} \tag{1}$$

where R is a planar region and Γ is the boundary of R. In the book in §8.8, we examine this problem for $R = \{(x,y) \mid 0 < x, y < 1\}$.

For an integer $N > 1$, we introduce a mesh size $h = 1/N$. With it, we define a rectangular mesh on $\overline{R} = R \cup \Gamma$:

$$(x_j, y_k) = \left(\tfrac{j}{N}, \tfrac{k}{N}\right), \qquad j,k = 0,1,...,N$$

At mesh points (=grid points) inside of R, we approximate the equation $\Delta u = g$. To do so, we recall the approximation

$$G''(x) = \frac{G(x+h) - 2G(x) + G(x-h)}{h^2} - \frac{h^2}{12} G^{(4)}(\xi)$$

with some $\xi \in [x-h, x+h]$. This assumes G is four times continuously differentiable on $[x-h, x+h]$.

Since

$$\Delta u(x,y) = \frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2}$$

we obtain at each $(x_j, y_k) \in R$ that

$$\begin{aligned} &\frac{u(x_{j+1},y_k) - 2u(x_j,y_k) + u(x_{j-1},y_k)}{h^2} \\ &\quad + \frac{u(x_j,y_{k+1}) - 2u(x_j,y_k) + u(x_j,y_{k-1})}{h^2} \\ &\quad - \frac{h^2}{12}\left[\frac{\partial^4 u(\xi_j,y_k)}{\partial x^4} + \frac{\partial^4 u(x_j,\eta_k)}{\partial y^4}\right] \\ &\quad = g(x_j,y_k) \end{aligned} \tag{2}$$

with $\xi_j \in \Big[x_{j-1}, x_{j+1}\Big]$, $\eta_k \in [y_{k-1}, y_{k+1}]$. We can delete the error terms involving the fourth derivatives of u. This leads to a new set of equations for approximating unknown $u_h \approx u$:

$$\Big\{u_h(x_j,y_k) \mid 1 < j,k < N\Big\}$$

The values of u_h at boundary mesh points on Γ are given by

$$u_h(x_j,y_k) = f(x_j,y_k), \qquad (x_j,y_k) \in \Gamma \tag{3}$$

This leads to the linear system

$$\frac{u_h(x_{j+1}, y_k) - 2u_h(x_j, y_k) + u_h(x_{j-1}, y_k)}{h^2}$$
$$+\frac{u_h(x_j, y_{k+1}) - 2u_h(x_j, y_k) + u_h(x_j, y_{k-1})}{h^2}$$
$$= g(x_j, y_k), \qquad (x_j, y_k) \in R$$

$$u_h(x_j, y_k) = f(x_j, y_k), \qquad (x_j, y_k) \in \Gamma \tag{4}$$

For interior mesh points, we can simplify this to

$$u_h(x_j, y_{k-1}) + u_h(x_{j-1}, y_k) - 4u_h(x_j, y_k)$$
$$+u_h(x_{j+1}, y_k) + u_h(x_j, y_{k+1}) = h^2 g(x_j, y_k)$$
$$(x_j, y_k) \in R \tag{5}$$

Thus we have a system of $(N+1)^2$ linear equations in the $(N+1)^2$ unknowns

$$\Big\{u_h(x_j, y_k) \mid 1 \le j, k \le N\Big\}$$

If $u(x, y)$ is four times continuously differentiable over $\overline{R}$, it can be shown that for some c,

$$\max_R \Big|u(x_j, y_k) - u_h(x_j, y_k)\Big| \le c\, h^2$$

JACOBI ITERATION

For $m = 0, 1, 2, ...,$ define

$$u_h^{(m+1)}(x_j, y_k) = \frac{1}{4}\Big\{u_h^{(m)}(x_j, y_{k-1}) + u_h^{(m)}(x_{j-1}, y_k) + u_h^{(m)}(x_{j+1}, y_k) + u_h^{(m)}(x_j, y_{k+1}) - h^2 g(x_j, y_k)\Big\}$$
$$(x_j, y_k) \in R \tag{6}$$

For values of $u_h^{(m)}$ at boundary points, use the given boundary conditions as in (3).

We can write the iteration in matrix-vector form as

$$\mathbf{u}_h^{(m+1)} = Q\mathbf{u}_h^{(m)} + \mathbf{G}_h$$

$$Q = \frac{1}{4}\begin{bmatrix} T & I & 0 & \cdots & 0 & 0 \\ I & T & I & & & \\ 0 & I & T & I & & \vdots \\ \vdots & & & \ddots & & \\ 0 & & & I & T & I \\ 0 & & \cdots & & I & T \end{bmatrix}$$

with I the identity of order $N-1$,

$$T = \begin{bmatrix} 0 & 1 & \cdots & & 0 \\ 1 & 0 & 1 & & \\ \vdots & & \ddots & & \vdots \\ & & 1 & 0 & 1 \\ 0 & & \cdots & 1 & 0 \end{bmatrix}$$

$$\mathbf{u}_h^{(m)} = \begin{bmatrix} \mathbf{u}_h^{(m)}(*, y_1) \\ \mathbf{u}_h^{(m)}(*, y_2) \\ \vdots \\ \mathbf{u}_h^{(m)}(*, y_{N-1}) \end{bmatrix}$$

$$\mathbf{u}_h^{(m)}(*, y_k) = \begin{bmatrix} u_h^{(m)}(x_1, y_k) \\ u_h^{(m)}(x_2, y_k) \\ \vdots \\ u_h^{(m)}(x_{N-1}, y_k) \end{bmatrix}$$

$$\mathbf{G}_h = \begin{bmatrix} \frac{1}{4}\left\{\mathbf{B}(y_1) - h^2\mathbf{G}_h(*, y_1)\right\} \\ \frac{1}{4}\left\{\mathbf{B}(y_2) - h^2\mathbf{G}_h(*, y_2)\right\} \\ \vdots \\ \frac{1}{4}\left\{\mathbf{B}(y_{N-1}) - h^2\mathbf{G}_h(*, y_{N-1})\right\} \end{bmatrix}$$

$$\mathbf{B}(y_k) = \begin{bmatrix} f(x_0, y_k) \\ 0 \\ \vdots \\ 0 \\ f(x_N, y_k) \end{bmatrix}$$

$$\mathbf{G}_h(*, y_k) = \begin{bmatrix} g(x_1, y_k) \\ g(x_2, y_k) \\ \vdots \\ g(x_{N-1}, y_k) \end{bmatrix}$$

The original discretization of (4)-(5) can be rewritten as

$$\mathbf{u}_h = Q\mathbf{u}_h + \mathbf{G}_h$$

with $\mathbf{u}_h$ written in accordance with the above definition of $\mathbf{u}_h^{(m)}$.

$$\mathbf{u}_h = Q\mathbf{u}_h + \mathbf{G}_h \tag{7}$$

$$\mathbf{u}_h^{(m+1)} = Q\mathbf{u}_h^{(m)} + \mathbf{G}_h \tag{8}$$

In this block matrix description of the system being solved, each block $\mathbf{u}_h(*, y_k)$ corresponds to the horizontal row of interior grid points at $y = y_k$, with the entries in $\mathbf{u}_h(*, y_k)$ being ordered in correspondence to the grid points being numbered from left to right in the grid. Note that Q is a symmetric matrix of order $(N-1)^2$.

If we write our linear system (7) in the form $Ax = b$, we have $A = I - Q$, again a symmetric matrix. With reference to the Gauss-Seidel iteration, note that the diagonal elements of A are positive.

For the iteration error $\mathbf{e}_h^{(m)}$ in the Jacobi iteration, subtract (8) from (7), obtaining

$$\mathbf{e}_h^{(m+1)} = Q\mathbf{e}_h^{(m)}, \qquad m = 0, 1, ... \tag{9}$$

The matrix Q is the earlier matrix M of our general framework in the case of the Gauss-Jacobi iteration. Thus we have convergence of the Jacobi iteration if and only if

$$r_\sigma(Q) < 1$$

Note in this case that

$$\|Q\|_1 = \|Q\|_\infty = 1$$

Thus we are driven to looking at $r_\sigma(Q)$, and in this case $\|Q\|_2 = r_\sigma(Q)$.

GAUSS-SEIDEL ITERATION

For $m = 0, 1, 2, ...,$ define

$$u_h^{(m+1)}(x_j, y_k) = \frac{1}{4}\Big\{u_h^{(m+1)}(x_j, y_{k-1}) + u_h^{(m+1)}(x_{j-1}, y_k) + u_h^{(m)}(x_{j+1}, y_k) + u_h^{(m)}(x_j, y_{k+1})\Big\} - h^2 g(x_j, y_k), \quad (x_j, y_k) \in R \tag{10}$$

For values of $u_h^{(m)}$ at boundary points, use the given boundary conditions as in (3).

We can use the above matrices to also write this iteration in matrix-vector format. In the language of the general framework given earlier, it amounts to letting N be the lower triangular portion of $A = I - Q$ and $P = N - A$. Then write

$$N\mathbf{u}_h^{(m+1)} = \mathbf{G}_h + P\mathbf{u}_h^{(m)}$$

RATES OF CONVERGENCE

Introduce

$$\xi = 1 - 2\sin^2\left(\frac{\pi}{2N}\right) \tag{11}$$

$$\omega^* = \frac{2}{1 + \text{sqrt}\left(1 - \xi^2\right)}$$

Then for the rates of convergence of the various iteration methods we have studied when applied to solving (7), $\mathbf{u}_h = Q\mathbf{u}_h + \mathbf{G}_h$, we have

Rate of Convergence	
Method	$r_\sigma(M)$
Jacobi	ξ
Gauss-Seidel	ξ^2
SOR	$\omega^* - 1$

THE RATE OF CONVERGENCE OF SOR

In the language of the general framework given earlier, the error equation for the SOR method will be

$$\mathbf{e}_h^{(m+1)} = M(\omega)\,\mathbf{e}_h^{(m)}, \qquad m = 0, 1, \ldots \tag{12}$$

The eigenvalues of $M(\omega)$ can be computed explicitly in this case, yielding

$$r_\sigma(M(\omega)) = \begin{cases} \dfrac{\omega\xi}{2} + \text{sqrt}\left(\left(\dfrac{\omega\xi}{2}\right)^2 + 4(1-\omega)\right), \\ \qquad\qquad 0 \le \omega \le \omega^* \\ \omega - 1, \\ \qquad\qquad \omega^* \le \omega \le 2 \end{cases}$$

We give graphs for a few cases of the discretization parameter N.

COSTS

Recall the notation from §8.7 regarding costs:

$$\text{Cost}(c, \varepsilon, n) = m^* \nu(n)$$

with c the factor by which the error was decreasing per iterate, n the order of the system, ε the error tolerance in

$$\left\| x - x^{(m)} \right\| \leq \varepsilon \left\| x - x^{(0)} \right\|$$

$\nu(n)$ the number of arithmetic operations per iterate calculation, and

$$m^* = \frac{-\log \varepsilon}{R(c)}, \qquad R(c) = -\log c$$

For $c = 1 - \delta$ with δ small, we have

$$R(c) \doteq \delta \qquad c = 1 - \delta, \qquad \delta \approx 0$$

Note that for Poisson example, $\nu(n) \approx 5n$, and thus the total cost of attaining the desired accuracy is

$$\text{Cost}(c, \varepsilon, n) = 5nm^*$$

APPLICATION TO SOLVING POISSON'S EQUATION

Gauss-Jacobi :

$$c = \xi = 1 - \tfrac{1}{2}\pi^2 h^2 + O\left(h^4\right)$$
$$\delta \approx \tfrac{1}{2}\pi^2 h^2$$

Gauss-Seidel :

$$c = \xi^2 = 1 - \pi^2 h^2 + O\left(h^4\right)$$
$$\delta \approx \pi^2 h^2$$

SOR :

$$c = \omega^* - 1 = 1 - 2\pi h + O\left(h^2\right)$$
$$\delta \approx 2\pi h$$

The improvement in the needed numbers of iterates is as follows, along with total costs for a desired accuracy.

Gauss-Jacobi vs. Gauss-Seidel : Gauss-Seidel requires approximately half the iterates required of Gauss-Jacobi. For Gauss-Seidel iteration,

$$\begin{aligned} \text{Cost}(c, \varepsilon, n) &\approx -\frac{5n \log \varepsilon}{\pi^2 h^2} = -\frac{5(N-1)^2 \log \varepsilon}{\pi^2 (N-1)^{-2}} \\ &\approx -\frac{5N^4 \log \varepsilon}{\pi^2} \end{aligned}$$

Gauss-Seidel vs. SOR : The factor giving the decrease in the number of iterates to be computed is

$$\frac{2\pi h}{\pi^2 h^2} = \frac{2}{\pi h}$$

Thus you are increasingly better off as $h \longrightarrow 0$. For the cost of SOR,

$$\begin{aligned} \text{Cost}(c, \varepsilon, n) &\approx -\frac{5n \log \varepsilon}{2\pi h} = -\frac{5(N-1)^2 \log \varepsilon}{2\pi (N-1)^{-1}} \\ &\approx -\frac{5N^3 \log \varepsilon}{2\pi} \end{aligned}$$

The costs increase rapidly with N.

Let K_J, K_{GS}, K_{SOR} denote the number of iterations needed to reduce the iteration error by a factor of 10, for the Jacobi, Gauss-Seidel, and SOR methods, respectively. The following table shows the results for several values of N.

	$N = 10$	$N = 20$	$N = 40$	$N = 80$
ξ	.9510565	.9876883	.9969173	.99922904
ξ^2	.9045085	.9755283	.9938442	.99845867
$\omega^* - 1$	.5278640	.7294538	.8544978	.92444658
K_J	46	186	746	2985
K_{GS}	23	93	373	1493
K_{SOR}	4	8	15	31

AN INITIAL GUESS

How to generate an initial guess $u_h^{(0)}$? Since we know the solution on the boundary,

$$u_h(x_j, y_k) = f(x_j, y_k), \qquad (x_j, y_k) \in \Gamma$$

We use an "interpolant" of this data to extend the boundary values to all of R. Define

$$\begin{aligned} u_h^{(0)}(x,y) \;=\; & (1-x)f(0,y) + xf(1,y) \\ & +(1-y)f(x,0) + yf(x,1) \\ & -[(1-y)(1-x)f(0,0) + xyf(1,1) \\ & +(1-y)xf(1,0) + y(1-x)f(0,1)] \end{aligned} \tag{13}$$

LINE ITERATION

Recall the discretization of (8.8.5):

$$\begin{aligned} u_h(x_j, y_k) \; &= \tfrac{1}{4}\Big\{u_h(x_{j+1}, y_k) + u_h(x_j, y_{k+1}) \\ &\quad + u_h(x_{j-1}, y_k) + u_h(x_j, y_{k-1})\Big\} \\ &\quad - \tfrac{h^2}{4} g(x_j, y_k), \quad 1 \le j, k \le N-1 \end{aligned}$$

Previously we solved individually for a new value of $u_h(x_j, y_k)$. Consider solving simultaneously for all the values in a row of the grid, say in row $\#k$.

The line Jacobi method is defined by

$$\begin{aligned} u_h^{(m+1)}(x_j, y_k) = \tfrac{1}{4}\Big\{ & u_h^{(m+1)}(x_{j+1}, y_k) \\ & + u_h^{(m+1)}(x_{j-1}, y_k) + u_h^{(m)}(x_j, y_{k+1}) \\ & + u_h^{(m)}(x_j, y_{k-1})\Big\} - \tfrac{h^2}{4} g(x_j, y_k), \quad 1 \le j \le N-1 \end{aligned}$$

for $k = 1, ..., N-1$. This is a tridiagonal system of order $N-1$; and it can be solved in approximately $5N$ arithmetic operations.

MULTIGRID ITERATION

I just give the general philosophy, as the details are quite complicated.

Imagine having a sequence of linear systems

$$A_N \mathbf{u}_N = \mathbf{b}_N, \quad N = N_0, N_1, N_2, ..., N_q$$

In our case with the Poisson equation, we would generally have $N_{j+1} = 4N_j$, approximately. We want to solve for the most accurate discretization, that for $N = N_q$.

The general philosophy is to use information from each of the systems $A_N \mathbf{u}_N = \mathbf{b}_N$ to solve all the systems with a finer mesh. There are a number of ways is which this may be carried out.

Step 1. Use the Jacobi method to iterate in $A_{N_q}\mathbf{u}_{N_q} = \mathbf{b}_{N_q}$, calculating a certain number of iterates (say K), to get $\mathbf{v} = \mathbf{u}_{N_q}^{(K)}$. Then calculate the residual

$$\mathbf{R} = \mathbf{b}_{N_q} - A_{N_q}\mathbf{v} = A_{N_q}\left(\mathbf{u}_{N_q} - \mathbf{v}\right)$$

Step 2. Find an approximate solution by performing the multigrid iteration on the system

$$A_{N_{q-1}}\delta = \mathcal{R}_{q,q-1}\mathbf{R}$$

calling the approximate solution $\widetilde{\delta}$.

Step 3. *Prolong* the solution $\widetilde{\delta}$ to a vector $\widehat{\delta}$ be of length N_q and then define the new iterate for $A_{N_q}\mathbf{u}_{N_q} = \mathbf{b}_{N_q}$ to be

$$\mathbf{u}_{N_q} \approx \mathbf{v} + \widehat{\delta}$$

This converges extremely rapidly; and there are public domain packages available which implement it for common classes of partial differential equations. In general these are the fastest means of solving discretizations of PDEs.

DIRECT METHODS

With the systems for some special PDEs, including the Poisson equation on a rectangle, it is possible to give an exact method of solution which is also extremely efficient. It depends on knowing the eigenvalues and eigenvectors for the matrix A_N. Since the matrix is symmetric and positive definite, there is a basis of orthogonal eigenvectors; and it can be found explicitly. Suppose it is written as

$$\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, ..., \mathbf{v}^{(N)}$$

with eigenvalues $\lambda_1, ..., \lambda_N$. Then write

$$\mathbf{b} = \sum_{j=1}^{N} \left(\mathbf{b}, \mathbf{v}^{(j)}\right) \mathbf{v}^{(j)}$$

The solution of $A\mathbf{u}_N = \mathbf{b}$ is given by

$$\mathbf{u}_N = \sum_{j=1}^{N} \frac{1}{\lambda_j} \left(\mathbf{b}, \mathbf{v}^{(j)}\right) \mathbf{v}^{(j)}$$

In the particular case of the Poisson equation over a rectangle, this can be computed in $O(N \log N)$ operations.

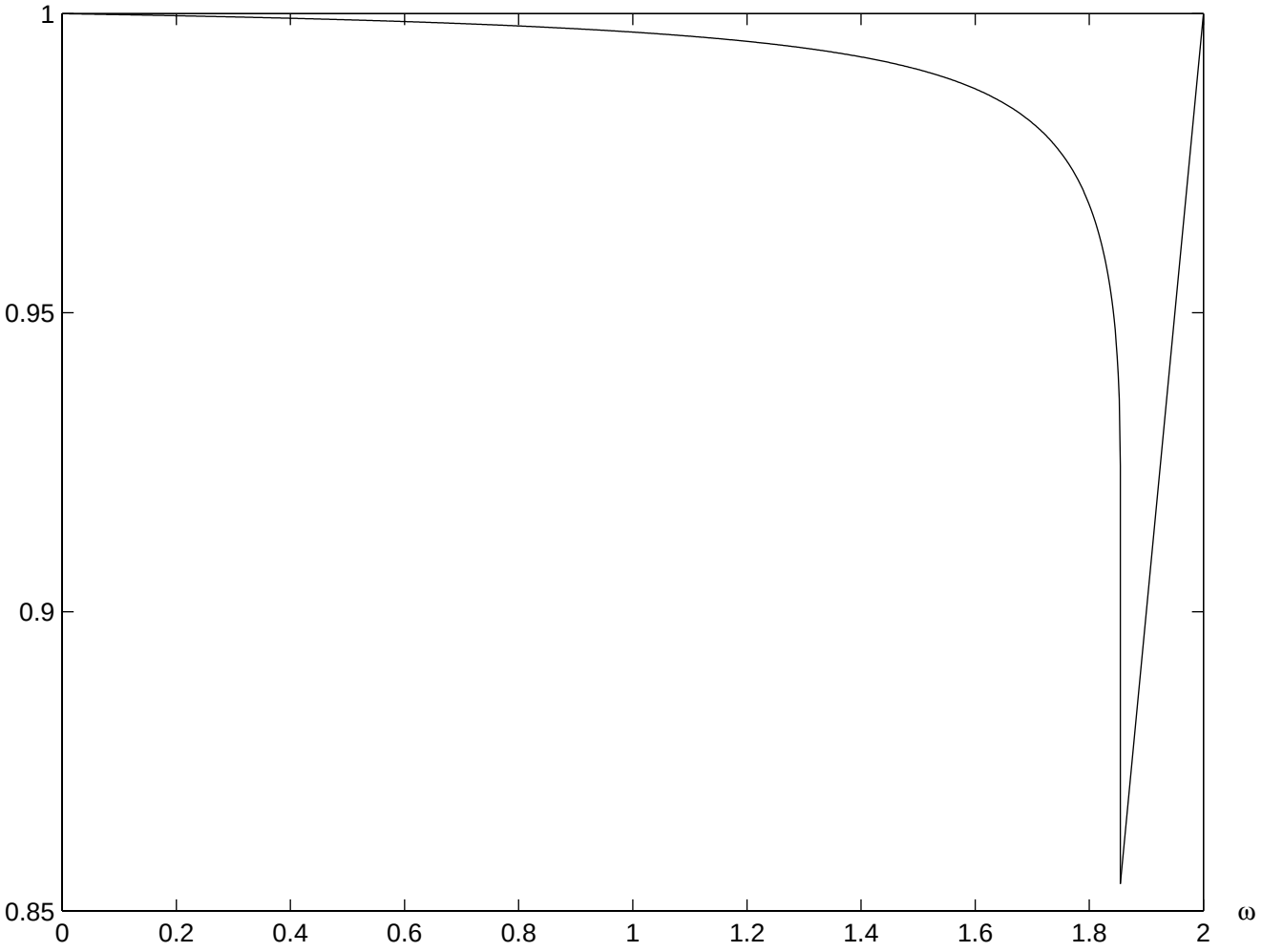

Figure 1: Convergence rate $r_\sigma(M(\omega))$ with $N = 40$ and $\omega^* = 1.85450$

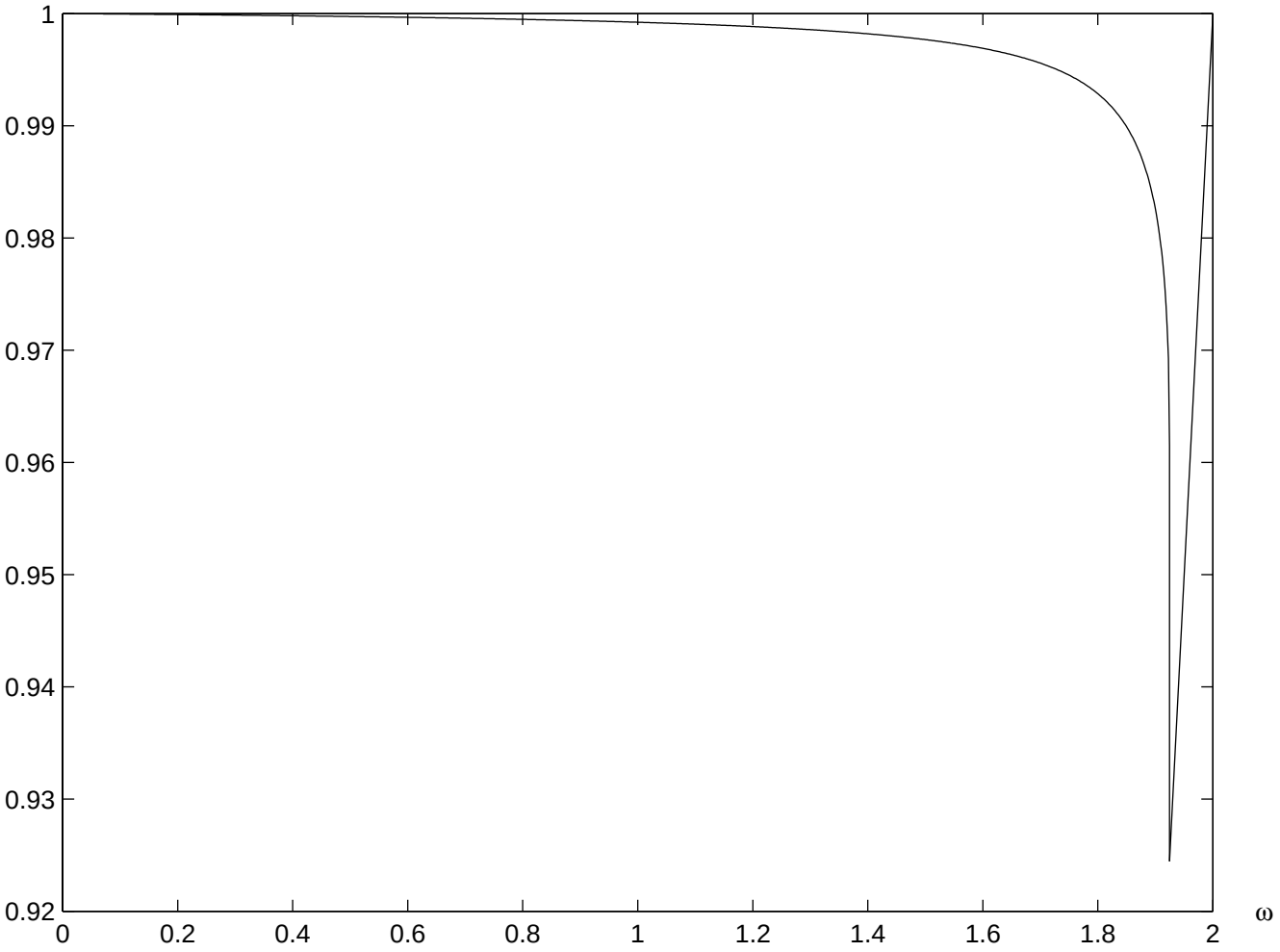

Figure 2: Convergence rate $r_\sigma(M(\omega))$ with $N = 80$ and $\omega^* = 1.92445$