Resumé of lecture 1

Driven Cavity, with $u = \sin^2 \pi x$ on top.

Resumé of lecture 1

Driven Cavity, with $u = \sin^2 \pi x$ on top.
Streamfunction-vorticity formulation:

Resumé of lecture 1

Driven Cavity, with $u = \sin^2 \pi x$ on top.
Streamfunction-vorticity formulation:
1. At each t given ω, find ψ:

$$\nabla^2 \psi = -\omega$$

with $\psi = 0$ all sides.

Resumé of lecture 1

Driven Cavity, with $u = \sin^2 \pi x$ on top.
Streamfunction-vorticity formulation:
1. At each t given ω, find ψ:

$$\nabla^2 \psi = -\omega$$

with $\psi = 0$ all sides.
2. With ω and now ψ known at t, find ω at $t + \Delta t$:

$$\frac{\partial \omega}{\partial t} = -\frac{\partial(\psi, \omega)}{\partial(x, y)} + \frac{1}{Re} \nabla^2 \omega$$

with ω on boundary so $\frac{\partial \psi}{\partial n}$ correct

Resumé of lecture 1

Driven Cavity, with $u = \sin^2 \pi x$ on top.
Streamfunction-vorticity formulation:
1. At each t given ω, find ψ:

$$\nabla^2 \psi = -\omega$$

with $\psi = 0$ all sides.
2. With ω and now ψ known at t, find ω at $t + \Delta t$:

$$\frac{\partial \omega}{\partial t} = -\frac{\partial(\psi, \omega)}{\partial(x, y)} + \frac{1}{Re} \nabla^2 \omega$$

with ω on boundary so $\frac{\partial \psi}{\partial n}$ correct

Physics of the Navier-Stokes equation, corner singularity, non-dimensional, classification PDEs, proper IC/BC

Resumé of lecture 1

Driven Cavity, with $u = \sin^2 \pi x$ on top.
Streamfunction-vorticity formulation:
1. At each t given ω, find ψ:

$$\nabla^2 \psi = -\omega$$

with $\psi = 0$ all sides.
2. With ω and now ψ known at t, find ω at $t + \Delta t$:

$$\frac{\partial \omega}{\partial t} = -\frac{\partial(\psi, \omega)}{\partial(x, y)} + \frac{1}{Re} \nabla^2 \omega$$

with ω on boundary so $\frac{\partial \psi}{\partial n}$ correct

Physics of the Navier-Stokes equation, corner singularity, non-dimensional, classification PDEs, proper IC/BC

Attempting numerical solution reveals poor understanding of question (physics and maths).

2.2 Finite differences – simple

Later, Part II on more sophisticated finite differences, as well as finite elements and spectral representation.

2.2 Finite differences – simple

Later, Part II on more sophisticated finite differences, as well as finite elements and spectral representation.

Finite computer $\rightarrow$ finite representation: spot data

2.2 Finite differences – simple

Later, Part II on more sophisticated finite differences, as well as finite elements and spectral representation.

Finite computer $\quad\rightarrow\quad$ finite representation: spot data

$$\omega_{ij}^n \approx \omega(x = i\Delta x, y = j\Delta x, t = n\Delta t).$$

for $i = 0, 1, \ldots, N$, $j = 0, 1, \ldots, N$ and $n = 0, 1, 2 \ldots$

Square mesh with $\Delta y = \Delta x$.

Approximation of derivatives

Forward differencing $\quad f_i' = \quad \dfrac{f_{i+1} - f_i}{\Delta x} + O(\Delta x)$

Approximation of derivatives

$$\text{Forward differencing} \quad f_i' = \frac{f_{i+1} - f_i}{\Delta x} + O(\Delta x)$$

$$\text{Backward differencing} \quad f_i' = \frac{f_i - f_{i-1}}{\Delta x} - O(\Delta x)$$

Approximation of derivatives

$$\begin{aligned}
\text{Forward differencing} \quad f_i' &= \frac{f_{i+1} - f_i}{\Delta x} + O(\Delta x) \\
\text{Backward differencing} \quad f_i' &= \frac{f_i - f_{i-1}}{\Delta x} - O(\Delta x) \\
\text{Central differencing} \quad f_i' &= \frac{f_{i+1} - f_{i-1}}{2\Delta x} + O(\Delta x^2)
\end{aligned}$$

Approximation of derivatives

$$
\begin{aligned}
\text{Forward differencing} \quad f_i' &= \frac{f_{i+1} - f_i}{\Delta x} + O(\Delta x) \\
\text{Backward differencing} \quad f_i' &= \frac{f_i - f_{i-1}}{\Delta x} - O(\Delta x) \\
\text{Central differencing} \quad f_i' &= \frac{f_{i+1} - f_{i-1}}{2\Delta x} + O(\Delta x^2)
\end{aligned}
$$

Curvature error cancels in central difference

Second derivative f''

$$f''_i \approx \frac{\left(f'_{i+\frac{1}{2}} \approx \frac{f_{i+1} - f_i}{\Delta x}\right) - \left(f'_{i-\frac{1}{2}} \approx \frac{f_i - f_{i-1}}{\Delta x}\right)}{\Delta x}$$

Second derivative f''

$$
\begin{aligned}
{f''}_i \approx & \quad \frac{\left(f'_{i+\frac{1}{2}} \approx \frac{f_{i+1} - f_i}{\Delta x}\right) - \left(f'_{i-\frac{1}{2}} \approx \frac{f_i - f_{i-1}}{\Delta x}\right)}{\Delta x} \\
= & \qquad \frac{f_{i+1} - 2f_i + f_{i-1}}{\Delta x^2} + O(\Delta x^2).
\end{aligned}
$$

Second derivative f''

$$
\begin{aligned}
f''_i \approx \quad & \frac{\left(f'_{i+\frac{1}{2}} \approx \frac{f_{i+1} - f_i}{\Delta x}\right) - \left(f'_{i-\frac{1}{2}} \approx \frac{f_i - f_{i-1}}{\Delta x}\right)}{\Delta x} \\
= \quad & \frac{f_{i+1} - 2f_i + f_{i-1}}{\Delta x^2} + O(\Delta x^2).
\end{aligned}
$$

Note

$$
f''_i \neq \left(f'_i\right)' = \frac{f_{i+2} - 2f_i + f_{i-2}}{4\Delta x^2}.
$$

Second derivative f''

$$
\begin{aligned}
{f''}_i \approx \quad & \frac{\left(f'_{i+\frac{1}{2}} \approx \dfrac{f_{i+1} - f_i}{\Delta x}\right) - \left(f'_{i-\frac{1}{2}} \approx \dfrac{f_i - f_{i-1}}{\Delta x}\right)}{\Delta x} \\
= \quad & \frac{f_{i+1} - 2f_i + f_{i-1}}{\Delta x^2} + O(\Delta x^2).
\end{aligned}
$$

Note

$$
{f''}_i \neq \left(f'_i\right)' = \frac{f_{i+2} - 2f_i + f_{i-2}}{4\Delta x^2}.
$$

– error 4 times as large.

Second derivative f''

$$f''_i \approx \frac{\left(f'_{i+\frac{1}{2}} \approx \frac{f_{i+1} - f_i}{\Delta x}\right) - \left(f'_{i-\frac{1}{2}} \approx \frac{f_i - f_{i-1}}{\Delta x}\right)}{\Delta x}$$

$$= \frac{f_{i+1} - 2f_i + f_{i-1}}{\Delta x^2} + O(\Delta x^2).$$

Note

$$f''_i \neq \left(f'_i\right)' = \frac{f_{i+2} - 2f_i + f_{i-2}}{4\Delta x^2}.$$

– error 4 times as large.

Also

$$(ab)'_i \neq a'_i b_i + a_i b'_i.$$

Local error analysis

by Taylor series

$$f_{i+1} \quad = \quad f(x = i\Delta x + \Delta x)$$

Local error analysis

by Taylor series

$$\begin{aligned} f_{i+1} &= f(x = i\Delta x + \Delta x) \\ &= f_i + \Delta x f_i' + \frac{1}{2}\Delta x^2 f_i'' + \frac{1}{6}\Delta x^3 f_i''' + \frac{1}{24}\Delta x^4 f_i'''' + \ldots \end{aligned}$$

Local error analysis

by Taylor series

$$
\begin{aligned}
f_{i+1} &= f(x = i\Delta x + \Delta x) \\
&= f_i + \Delta x f_i' + \tfrac{1}{2}\Delta x^2 f_i'' + \tfrac{1}{6}\Delta x^3 f_i''' + \tfrac{1}{24}\Delta x^4 f_i'''' + \dots
\end{aligned}
$$

Hence

$$
f_{i+1} - 2f_i + f_{i-1} = \Delta x^2 f_i'' + \tfrac{1}{12}\Delta x^4 f_i''''.
$$

Local error analysis

by Taylor series

$$\begin{aligned} f_{i+1} &= f(x = i\Delta x + \Delta x) \\ &= f_i + \Delta x f_i' + \tfrac{1}{2}\Delta x^2 f_i'' + \tfrac{1}{6}\Delta x^3 f_i''' + \tfrac{1}{24}\Delta x^4 f_i'''' + \dots \end{aligned}$$

Hence

$$f_{i+1} - 2f_i + f_{i-1} = \Delta x^2 f_i'' + \tfrac{1}{12}\Delta x^4 f_i''''.$$

Try to use central differences, so $O(\Delta x^2)$ in spatial differentiation.

Local error analysis

by Taylor series

$$\begin{aligned} f_{i+1} &= f(x = i\Delta x + \Delta x) \\ &= f_i + \Delta x f_i' + \tfrac{1}{2}\Delta x^2 f_i'' + \tfrac{1}{6}\Delta x^3 f_i''' + \tfrac{1}{24}\Delta x^4 f_i'''' + \dots \end{aligned}$$

Hence

$$f_{i+1} - 2f_i + f_{i-1} = \Delta x^2 f_i'' + \tfrac{1}{12}\Delta x^4 f_i''''.$$

Try to use central differences, so $O(\Delta x^2)$ in spatial differentiation.

Forward time differencing adequate for driven cavity – see later.

Laplacian

$$\left(\nabla^2\psi\right)_{ij} \approx \frac{\psi_{i+1\,j} - 2\psi_{i\,j} + \psi_{i-1\,j}}{\Delta x^2} + \frac{\psi_{i\,j+1} - 2\psi_{i\,j} + \psi_{i\,j-1}}{\Delta x^2},$$

Laplacian

$$\left(\nabla^2\psi\right)_{ij} \approx \frac{\psi_{i+1\,j} - 2\psi_{i\,j} + \psi_{i-1\,j}}{\Delta x^2} + \frac{\psi_{i\,j+1} - 2\psi_{i\,j} + \psi_{i\,j-1}}{\Delta x^2},$$

written with a 'numerical molecule'

$$\approx \frac{1}{\Delta x^2}\begin{pmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{pmatrix}\psi_{ij}.$$

2.3 Poisson problem: $\nabla^2\psi = -\omega$

At interior points, $i = 1 \to N-1, j = 1 \to N-1$, solve

$$\frac{1}{\Delta x^2}\begin{pmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{pmatrix}\psi_{ij} = -\omega_{ij},$$

2.3 Poisson problem: $\nabla^2\psi = -\omega$

At interior points, $i = 1 \rightarrow N-1, j = 1 \rightarrow N-1$, solve

$$\frac{1}{\Delta x^2}\begin{pmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{pmatrix}\psi_{ij} = -\omega_{ij},$$

with boundary conditions

$$\psi = 0 \text{ for } i = 0 \;\&\; N,\ j = 0 \rightarrow N \text{ and for } j = 0 \;\&\; N,\ i = 0 \rightarrow N.$$

2.3 Poisson problem: $\nabla^2\psi = -\omega$

At interior points, $i = 1 \to N - 1, j = 1 \to N - 1$, solve

$$\frac{1}{\Delta x^2} \begin{pmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{pmatrix} \psi_{ij} = -\omega_{ij},$$

with boundary conditions

$$\psi = 0 \text{ for } i = 0 \;\&\; N,\ j = 0 \to N \text{ and for } j = 0 \;\&\; N,\ i = 0 \to N.$$

Large problem in linear algebra

2.3 Poisson problem: $\nabla^2\psi = -\omega$

At interior points, $i = 1 \to N-1, j = 1 \to N-1$, solve

$$\frac{1}{\Delta x^2}\begin{pmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{pmatrix}\psi_{ij} = -\omega_{ij},$$

with boundary conditions

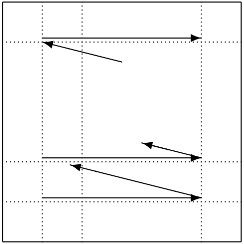
$$\psi = 0 \text{ for } i = 0 \ \& \ N,\ j = 0 \to N \text{ and for } j = 0 \ \& \ N,\ i = 0 \to N.$$

Large problem in linear algebra
90% CPU of most programs – worth a good method

Simplest – Gauss-Seidel

Sweep through interior

$$\begin{aligned} j = 1 : &\quad i = 1 \to N-1 \\ j = 2 : &\quad i = 1 \to N-1 \\ &\downarrow \\ j = N-1 : &\quad i = 1 \to N-1 \end{aligned}$$

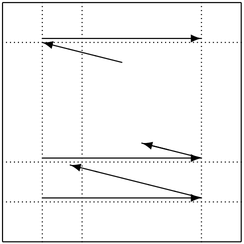


and then repeat.

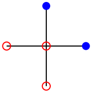
Simplest – Gauss-Seidel

Sweep through interior

$$
\begin{aligned}
j = 1 : &\quad i = 1 \to N - 1 \\
j = 2 : &\quad i = 1 \to N - 1 \\
&\downarrow \\
j = N - 1 : &\quad i = 1 \to N - 1
\end{aligned}
$$



and then repeat.

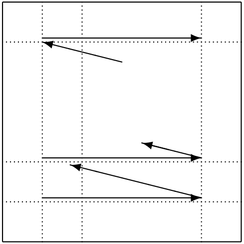


$$
\psi^{\mathrm{new}}_{i\,j} = \frac{1}{4}\left(\psi^{\mathrm{old}}_{i+1\,j} + \psi^{\mathrm{new}}_{i-1\,j} + \psi^{\mathrm{old}}_{i\,j+1} + \psi^{\mathrm{new}}_{i\,j-1} + \Delta x^2 \omega_{ij}\right).
$$

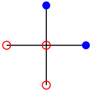
Simplest – Gauss-Seidel

Sweep through interior

$$\begin{aligned} j = 1: &\quad i = 1 \to N-1 \\ j = 2: &\quad i = 1 \to N-1 \\ &\downarrow \\ j = N-1: &\quad i = 1 \to N-1 \end{aligned}$$



and then repeat.



$$\psi_{i\,j}^{\text{new}} = \frac{1}{4}\left(\psi_{i+1\,j}^{\text{old}} + \psi_{i-1\,j}^{\text{new}} + \psi_{i\,j+1}^{\text{old}} + \psi_{i\,j-1}^{\text{new}} + \Delta x^2 \omega_{ij}\right).$$

To converge need $O(N^2)$ iterations/compete sweeps $\to O(N^4)$ operations.

A little better – Successive Over Relaxation

$$\psi_{ij}^{\mathrm{new}} = (1-r)\psi_{ij}^{\mathrm{old}} + r\{\text{above expression for } \psi_{ij}^{\mathrm{new}}\}.$$

A little better – Successive Over Relaxation

$$\psi_{ij}^{\text{new}} = (1-r)\psi_{ij}^{\text{old}} + r\{\text{above expression for } \psi_{ij}^{\text{new}}\}.$$

$$\begin{aligned} 0 < r < 1 &\quad \text{under-relax} \\ r = 1 &\quad \text{Gauss-Seidel} \\ 1 < r < 2 &\quad \text{over-relax} \\ r \geq 2 &\quad \text{unstable} \end{aligned}$$

A little better – Successive Over Relaxation

$$\psi_{ij}^{\text{new}} = (1-r)\psi_{ij}^{\text{old}} + r\{\text{above expression for } \psi_{ij}^{\text{new}}\}.$$

$$\begin{aligned} 0 < r < 1 &\quad \text{under-relax} \\ r = 1 &\quad \text{Gauss-Seidel} \\ 1 < r < 2 &\quad \text{over-relax} \\ r \geq 2 &\quad \text{unstable} \end{aligned}$$

Optimal (for this problem and large N)

$$r = \frac{2}{1 + \frac{\pi}{N}}.$$

A little better – Successive Over Relaxation

$$\psi_{ij}^{\text{new}} = (1-r)\psi_{ij}^{\text{old}} + r\{\text{above expression for } \psi_{ij}^{\text{new}}\}.$$

$$\begin{aligned} 0 < r < 1 &\quad \text{under-relax} \\ r = 1 &\quad \text{Gauss-Seidel} \\ 1 < r < 2 &\quad \text{over-relax} \\ r \geq 2 &\quad \text{unstable} \end{aligned}$$

Optimal (for this problem and large N)

$$r = \frac{2}{1 + \frac{\pi}{N}}.$$

With optimal r need $2N$ iterations for 4 figure accuracy
$\rightarrow$ total cost $O(N^3)$ operations.

2.4 Test code

1. $\omega = 0 \to \psi = 0$?

2.4 Test code

1. $\omega = 0 \rightarrow \psi = 0$?
    - Check loops – range-checking option of compiler

2.4 Test code

1. $\omega = 0 \rightarrow \psi = 0$?
   - Check loops – range-checking option of compiler
   - Compile of two types of machine – uninitialised variables

2.4 Test code

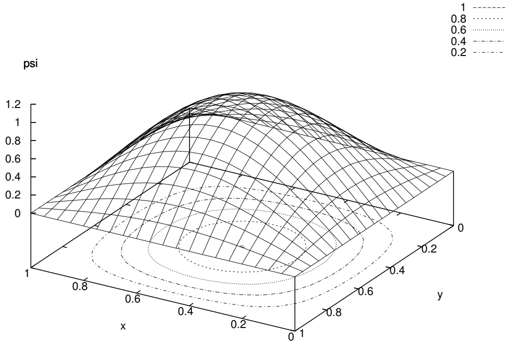
1. $\omega = 0 \rightarrow \psi = 0$?
   - Check loops – range-checking option of compiler
   - Compile of two types of machine – uninitialised variables
2. $\omega = 2\pi^2 \sin \pi x \sin \pi y \rightarrow \psi = \sin \pi x \sin \pi y$?

2.4 Test code

1. $\omega = 0 \rightarrow \psi = 0$?
   - Check loops – range-checking option of compiler
   - Compile of two types of machine – uninitialised variables
2. $\omega = 2\pi^2 \sin \pi x \sin \pi y \rightarrow \psi = \sin \pi x \sin \pi y$?
   - Plot $\psi(x, y)$ – shape OK? magnitude correct?

2.4 Test code

1. $\omega = 0 \rightarrow \psi = 0$?
   - Check loops – range-checking option of compiler
   - Compile of two types of machine – uninitialised variables
2. $\omega = 2\pi^2 \sin \pi x \sin \pi y \rightarrow \psi = \sin \pi x \sin \pi y$?
   - Plot $\psi(x, y)$ – shape OK? magnitude correct?

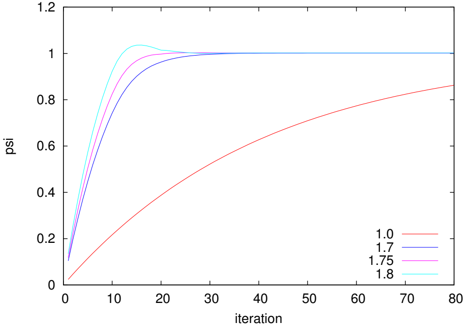


Test code 2

- $\psi(\frac{1}{2}, \frac{1}{2})$ vs number of iterations

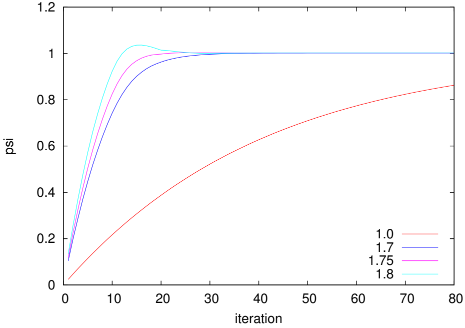
Test code 2

- $\psi(\frac{1}{2}, \frac{1}{2})$ vs number of iterations



Test code 2

- $\psi(\frac{1}{2}, \frac{1}{2})$ vs number of iterations



For $N = 20$ Gauss-Seidel needs 500 iterations, whereas SOR with optimal $r \approx 1.75$ needs 20.

Test code 3

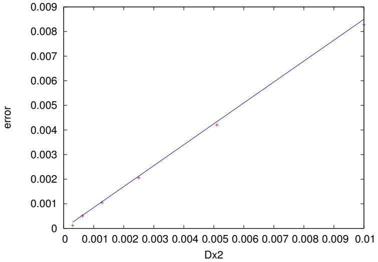
3. Variation with Δx of maximum error

$$\text{Error} = \max_{\text{grid}} \left| \psi_{ij}^{\text{numerical}} - \psi^{\text{theory}}(i\Delta x, j\Delta) \right| ,$$

Test code 3

3. Variation with Δx of maximum error

$$\text{Error} = \max_{\text{grid}} \left| \psi_{ij}^{\text{numerical}} - \psi^{\text{theory}}(i\Delta x, j\Delta) \right|,$$



Test code 4

For this test problem, max error $\approx 0.85 \Delta x^2$

Test code 4

For this test problem, max error $\approx 0.85\Delta x^2$
Hence

$$\begin{aligned} 1\% \text{ error (normal working)} \quad & \text{at } N = 10 \\ 10^{-3} \text{ error (if really needed)} \quad & \text{at } N = 28 \end{aligned}$$

Test code 4

For this test problem, max error $\approx 0.85\Delta x^2$
Hence

$$\begin{aligned} 1\% \text{ error (normal working)} &\quad \text{at } N = 10 \\ 10^{-3} \text{ error (if really needed)} &\quad \text{at } N = 28 \end{aligned}$$

But $\text{CPU}_{28} \approx 20\text{CPU}_{10}$

2.5 Code Quality

One-off code (written today, used today, never again): simple, clear layout, no tricks

2.5 Code Quality

One-off code (written today, used today, never again): simple, clear layout, no tricks

Production code:

2.5 Code Quality

One-off code (written today, used today, never again): simple, clear layout, no tricks

Production code:

- Comments on most lines

2.5 Code Quality

One-off code (written today, used today, never again): simple, clear layout, no tricks

Production code:

- Comments on most lines
- Test for problems, halt with helpful message

2.5 Code Quality

One-off code (written today, used today, never again): simple, clear layout, no tricks

Production code:

- Comments on most lines
- Test for problems, halt with helpful message
- Bullet-proof – no indirect action

2.5 Code Quality

One-off code (written today, used today, never again): simple, clear layout, no tricks

Production code:

- Comments on most lines
- Test for problems, halt with helpful message
- Bullet-proof – no indirect action
- Fast and efficient

2.5 Code Quality

One-off code (written today, used today, never again): simple, clear layout, no tricks

Production code:

- Comments on most lines
- Test for problems, halt with helpful message
- Bullet-proof – no indirect action
- Fast and efficient

EG avoid repeating same calculation, so first set $r1 = 1 - r$, $r025 = 0.25r$ and $h2w_{ij} = h^2\omega_{ij}$.

2.5 Code Quality

One-off code (written today, used today, never again): simple, clear layout, no tricks

Production code:

- Comments on most lines
- Test for problems, halt with helpful message
- Bullet-proof – no indirect action
- Fast and efficient

EG avoid repeating same calculation, so first set $r1 = 1 - r$, $r025 = 0.25r$ and $h2w_{ij} = h^2\omega_{ij}$. Then

$$\psi_{ij} = r1\psi_{ij} + r025\left[\begin{pmatrix} & 1 & \\ 1 & & 1 \\ & 1 & \end{pmatrix}\psi_{ij} + h2w_{ij}\right],$$

2.5 Code Quality

One-off code (written today, used today, never again): simple, clear layout, no tricks

Production code:

- Comments on most lines
- Test for problems, halt with helpful message
- Bullet-proof – no indirect action
- Fast and efficient

EG avoid repeating same calculation, so first set $r1 = 1 - r$, $r025 = 0.25r$ and $h2w_{ij} = h^2\omega_{ij}$. Then

$$\psi_{ij} = r1\psi_{ij} + r025\left[\begin{pmatrix} & 1 & \\ 1 & & 1 \\ & 1 & \end{pmatrix}\psi_{ij} + h2w_{ij}\right],$$

Packages: NAG, LAPACK, matrix routines

2.6 Simple graphing

Program writes out table: on ith line x_i, y_i, and z_i if contouring.

2.6 Simple graphing

Program writes out table: on ith line x_i, y_i, and z_i if contouring.

Pipe output to a results file $a.out \ > \ res$.

2.6 Simple graphing

Program writes out table: on *i*th line x_i, y_i, and z_i if contouring.

Pipe output to a results file $a.out \; > \; res$.

Public domain simple graphs *gnuplot*.

2.6 Simple graphing

Program writes out table: on ith line x_i, y_i, and z_i if contouring.

Pipe output to a results file $a.out\ >\ res$.

Public domain simple graphs *gnuplot*.

Line diagrams $y(x)$: $>$ *plot 'res' with lines*

2.6 Simple graphing

Program writes out table: on ith line x_i, y_i, and z_i if contouring.

Pipe output to a results file $a.out\ >\ res$.

Public domain simple graphs *gnuplot*.

Line diagrams $y(x)$: $>$ *plot 'res' with lines*

– (auto)scale, label, logs, multiple plots

2.6 Simple graphing

Program writes out table: on ith line x_i, y_i, and z_i if contouring.

Pipe output to a results file *a.out* $>$ *res*.

Public domain simple graphs *gnuplot*.

Line diagrams $y(x)$: $>$ *plot 'res' with lines*

– (auto)scale, label, logs, multiple plots

Contour plots $z(x, y)$: $>$ *splot 'res' w l*

2.6 Simple graphing

Program writes out table: on ith line x_i, y_i, and z_i if contouring.

Pipe output to a results file $a.out \ > \ res$.

Public domain simple graphs *gnuplot*.

Line diagrams $y(x)$: $>$ *plot 'res' with lines*

– (auto)scale, label, logs, multiple plots

Contour plots $z(x, y)$: $>$ *splot 'res' w l*

Many options: list with egs: $>$ *help*.

2.6 Simple graphing

Program writes out table: on *i*th line x_i, y_i, and z_i if contouring.

Pipe output to a results file *a.out* $>$ *res*.

Public domain simple graphs *gnuplot*.

Line diagrams $y(x)$: $>$ *plot 'res' with lines*

– (auto)scale, label, logs, multiple plots

Contour plots $z(x, y)$: $>$ *splot 'res' w l*

Many options: list with egs: $>$ *help*.
End: $>$ *quit*