1.723 - COMPUTATIONAL METHODS FOR FLOW IN POROUS MEDIA
Spring 2008

FOURIER SPECTRAL METHODS: MISCIBLE FLOW IN 2D. VISCOUS FINGERING

Luis Cueto-Felgueroso

1. MATHEMATICAL MODEL

1.1. Model equations

Consider the equations of two-dimensional horizontal miscible flow in a homogeneous porous medium. In non-dimensional form, and assuming an incompressible system, the concentration transport is modeled by

$$\frac{\partial c}{\partial t} + \boldsymbol{\nabla} \cdot \left(c\,\boldsymbol{u} - \frac{1}{Pe}\boldsymbol{\nabla} c \right) = 0 \tag{1}$$

where the Darcy velocity $\boldsymbol{u} = (u_x, u_y)$ is given in terms of the pressure p and the concentration-dependent viscosity $\mu(c)$, as

$$\boldsymbol{u} = -\frac{1}{\mu(c)}\boldsymbol{\nabla} p \tag{2}$$

For an incompressible system the continuity equation reduces to the constraint

$$\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0 \tag{3}$$

Finally, the viscosity is assumed to depend on the concentration of the mixture as

$$\mu = e^{-R\,c} \tag{4}$$

where R is the natural logarithm of the viscosity ratio. Following the standard, pressure-based approach, we could introduce (2) in (3) to arrive at the following elliptic equation for the pressure

$$\boldsymbol{\nabla} \cdot \left(-\frac{1}{\mu(c)}\boldsymbol{\nabla} p \right) = 0 \tag{5}$$

In the present case, on the other hand, we will be working with the *stream function-vorticity* formulation. Let us define the stream function, Ψ, such that

$$u_x = \frac{\partial \Psi}{\partial y} \qquad u_y = -\frac{\partial \Psi}{\partial x} \tag{6}$$

It follows from the above definition that the velocities given by (6) satisfy the incompressibility constraint, as

$$\boldsymbol{\nabla}\cdot\boldsymbol{u} = \frac{\partial}{\partial x}\left(\frac{\partial\Psi}{\partial y}\right) + \frac{\partial}{\partial y}\left(-\frac{\partial\Psi}{\partial x}\right) = 0 \tag{7}$$

The flow vorticity is defined as the vector $\boldsymbol{\omega} = \boldsymbol{\nabla}\times\boldsymbol{u}$; for 2D flows, $\boldsymbol{\omega} = \omega\,\boldsymbol{k}$, where

$$\omega = \frac{\partial u_y}{\partial x} - \frac{\partial u_x}{\partial y} \tag{8}$$

and hence we will loosely refer to vorticity as the vector $\omega\,\boldsymbol{k}$ or the scalar ω without distinction hereafter. The stream function and the vorticity are related by

$$\omega = \frac{\partial u_y}{\partial x} - \frac{\partial u_x}{\partial y} = \frac{\partial}{\partial x}\left(-\frac{\partial\Psi}{\partial x}\right) - \frac{\partial}{\partial y}\left(\frac{\partial\Psi}{\partial y}\right) = -\Delta\Psi \tag{9}$$

On the other hand, starting from the definition of the Darcy velocity we may arrive at the following expression for the vorticity

$$\boldsymbol{\omega} = \boldsymbol{\nabla}\times\boldsymbol{u} = \boldsymbol{\nabla}\times\left(-\frac{1}{\mu}\boldsymbol{\nabla}p\right) = -\boldsymbol{\nabla}\left(\frac{1}{\mu}\right)\times\boldsymbol{\nabla}p \tag{10}$$

which may be rearranged as

$$\boldsymbol{\omega} = \boldsymbol{\nabla}\left(\frac{1}{\mu}\right)\times(\mu\boldsymbol{u}) = \mu\frac{d}{dc}\left(\frac{1}{\mu}\right)\boldsymbol{\nabla}c\times\boldsymbol{u} \tag{11}$$

Finally, it follows from the exponential dependence of the viscosity on the concentration that

$$\frac{d}{dc}\left(\frac{1}{\mu}\right) = -\frac{\mu'}{\mu^2} = \frac{R}{\mu} \tag{12}$$

which reduces (10) to

$$\boldsymbol{\omega} = R\,\boldsymbol{\nabla}c\times\boldsymbol{u} \tag{13}$$

or

$$\omega = R\left(\frac{\partial c}{\partial x}u_y - \frac{\partial c}{\partial y}u_x\right) \tag{14}$$

Hence, instead of the solving for the pressure with (5), we may solve for the stream function through

$$\Delta\Psi = -\omega \qquad \omega = R\left(\frac{\partial c}{\partial x}u_y - \frac{\partial c}{\partial y}u_x\right) \tag{15}$$

1.2. Simulation set up: initial and boundary conditions

In summary, we will consider the problem

$$\begin{cases} \dfrac{\partial c}{\partial t} + \boldsymbol{\nabla} \cdot \left(c\,\boldsymbol{u} - \dfrac{1}{Pe} \boldsymbol{\nabla} c \right) = 0 \\ \Delta \Psi = -\omega \end{cases} \tag{16}$$

with

$$\omega = R \left(\frac{\partial c}{\partial x} u_y - \frac{\partial c}{\partial y} u_x \right) \tag{17}$$

and

$$u_x = \frac{\partial \Psi}{\partial y} \qquad u_y = -\frac{\partial \Psi}{\partial x} \tag{18}$$

The idealized initial concentration field corresponds to an infinite array of alternate solvent and oil strips. Assuming that the distances between solvent strips are large enough, we may thus enforce periodicity in both directions, x and y. This will allow us to study a "fully developed" flow, thus being able to characterize the dissipation properties of the system under reasonably homogeneous conditions in the streamwise direction. Initially, oil and solvent are moving with a constant, horizontal velocity $\boldsymbol{u}^0 = \left(u_x^0, u_y^0\right)$, compatible with a certain pressure gradient.

In order to allow for an efficient solution of the Laplacian, it is very convenient to be able to impose periodicity on the stream function as well. Thus, we will decompose the velocity field $\boldsymbol{u}$ into a base, constant flow (the initial flow $\boldsymbol{u}^0$), plus a periodic fluctuation $\widetilde{\boldsymbol{u}}$, as $\boldsymbol{u} = \boldsymbol{u}_0 + \widetilde{\boldsymbol{u}}$. With this decomposition in mind, let us decompose the stream function as

$$\Psi = \Psi^0 + \widetilde{\Psi} \tag{19}$$

where Ψ^0 generates the constant, base velocity, i.e. $\Psi^0 = u_x^0\, y - u_y^0\, x$, while the periodic part, $\widetilde{\Psi}$, is associated to the fluctuating velocity

$$\widetilde{u}_x = \frac{\partial \widetilde{\Psi}}{\partial y} \qquad \widetilde{u}_y = -\frac{\partial \widetilde{\Psi}}{\partial x} \tag{20}$$

With the above decomposition, we just need to solve for the fluctuating part of the stream function, as

$$\Delta \widetilde{\Psi} = -\omega \tag{21}$$

where the definition of the vorticity is the same as in (17). Once we have computed $\widetilde{\Psi}$ from (21), the total velocity is recovered as

$$\begin{aligned} u_x &= \frac{\partial \Psi}{\partial y} = u_x^0 + \frac{\partial \widetilde{\Psi}}{\partial y} \\ u_y &= -\frac{\partial \Psi}{\partial x} = u_y^0 - \frac{\partial \widetilde{\Psi}}{\partial x} \end{aligned} \tag{22}$$

2. NUMERICAL TECHNIQUES

The spatial discretization will be carried out using a Fourier pseudospectral (collocation) method. Thus, concentrations, velocities, vorticities and stream functions will be assumed to be periodic, and expanded in Fourier modes in the x and y directions.

Let us start in 1D with a given function $u(x)$, periodic in $[-\pi, \pi]$. Its Fourier series expansion, written in compact (complex) form, is

$$u(x) = \sum_{k=-\infty}^{\infty} \widehat{u}_k e^{ikx} \tag{23}$$

where the expansion coefficients are given by

$$\widehat{u}_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} u(x) e^{-ikx} dx \tag{24}$$

In practical computations we require *truncated* versions of the expansion (23), and a discretization of the transform to Fourier space (24). Thus, let us consider an N-term expansion of the form (23),

$$u(x) \approx \sum_{k=-N/2+1}^{N/2} \widehat{u}_k e^{ikx} \tag{25}$$

which implies a discretization with N *modes* in Fourier space, and a discretization in physical space (the interval $[-\pi, \pi]$) using N grid points and their associated grid function $\{u_j, j = 1, \ldots, N\}$. The continuous transform (24) may be turned into a discrete transform through the use of the nodes $\{x_j, j = 1, \ldots, N\}$ as quadrature points, as

$$\widehat{u}_k \approx \frac{\Delta x}{2\pi} \sum_{j=1}^{N} u_j e^{-ikx_j} \tag{26}$$

The efficiency and potential accuracy of the above Fourier series approximation is critically determined by how fast the expansion coefficients $\widehat{u}_k$ decay as k increases. The decay of these coefficients is, on the other hand, closely related to the smoothness of $u(x)$. Consider for example the function

$$u(x) = \frac{1-p^2}{4p} \left(\frac{1-p^2}{(1+p^2) - 2p\cos(x)} - \frac{1-p}{1+p} \right) \tag{27}$$

with $p < 1$ and $x \in [-\pi, \pi]$. Figure 1 plots a few members of this family of functions and their $N = 1024$ first Fourier coefficients, computed using the Fast Fourier Transform (FFT). In Matlab, this may be done with the following lines of code:

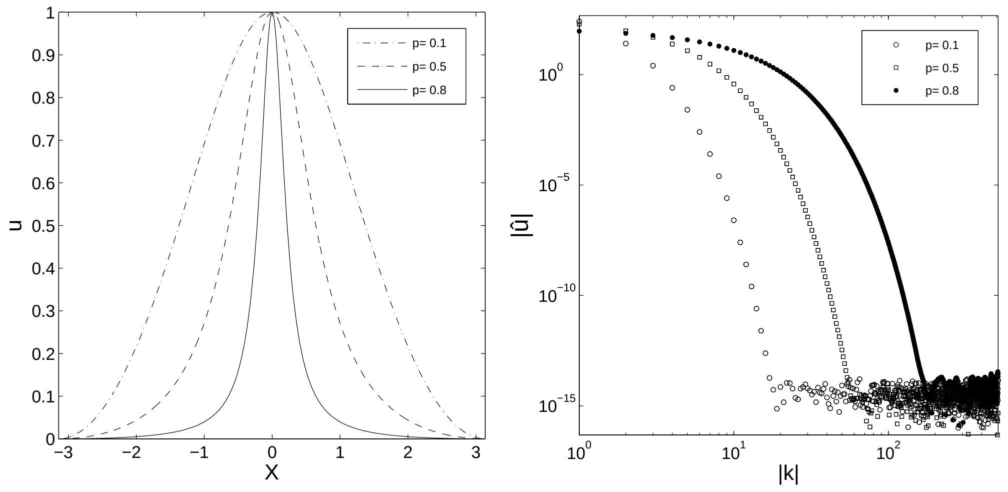


Figure 1. Sample functions from the family (27) (left) and their computed Fourier coefficients (right).

```
N   = 1024;
h   = 2*pi/N;
x= -pi:h:pi;x= x(1:N);
%Wave numbers...
k= [0:(N/2-1) (-N/2):(-1)];

ps= [0.1 0.5 0.8];
for ip= 1:length(ps);
    p= ps(ip);
    u= (1-pˆ2)./( (1+pˆ2) - 2*p*cos(x) );
    a= (1-p)/(1+p);
    b= (1-p*p)/(4*p);
    u= (u-a)*b;

    %Fourier coefficients
    uhat= fft(u);

    %Plot function
    figure(1);plot(x,u,’k’);axis tight;axis square;
    xlabel(’x’,’fontsize’,18);ylabel(’u’,’fontsize’,18);
    hold on

    %Plot Fourier coefficients
    figure(2);loglog(abs(k),abs(uhat),’k.’,’markersize’,12);axis square;
    xlabel(’|k|’,’fontsize’,18);ylabel(’|û|’,’fontsize’,20);axis tight
    hold on
end;
```

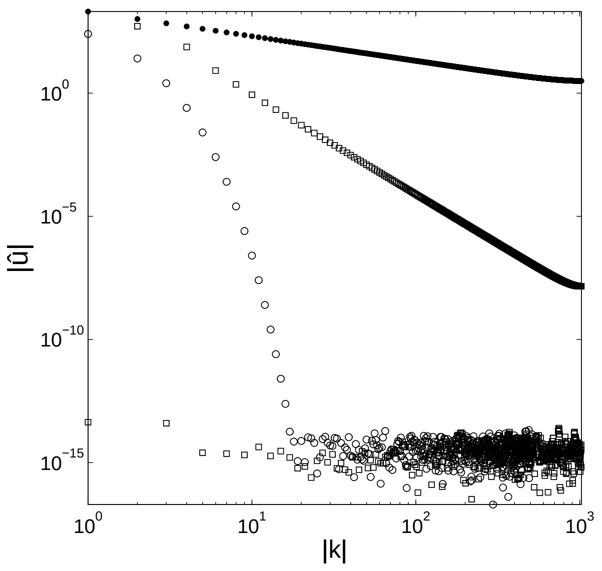


Figure 2. Decay of Fourier coefficients for functions with different smoothness.

Note the way in which the wave numbers k are defined in the code, `k= [0:(N/2-1) (-N/2):(-1)]`, which is the default wave number arrangement in Matlab's FFT. The sharper features of the function as $p \to 1$ have a clear impact on the decay of its expansion coefficients. For $p = 0.1$, the function is very well represented by just a few modes, while for $p = 0.8$ we require more than a hundred modes to achieve the same accuracy in the truncated Fourier expansion. Nevertheless, in all cases the Fourier coefficients exhibit an exponential decay as $k \to \infty$, and the resulting approximation will still be very competitive compared to approximations based on low order polynomials.

Let us now compare the function (27) ($p = 0.1$), which we will refer to as $f_1(x)$, with the functions $f_2(x) = |\sin(x)|^3$ and the "sawtooth" function, $f_3(x) = x$ (u periodic in $[-\pi, \pi]$). Their Fourier coefficients may be computed as exposed above, and the first 1024 of them are shown in figure 2. While for sufficiently smooth functions the use of Fourier expansions is an extremely powerful "data compression" technique, for non-smooth functions the decay of the expansion coefficients may be too slow for the method to be competitive with lower order methods.

2.1. Spectral differentiation

The m-th derivative of $u(x)$ can be approximated by the m-th derivative of its truncated Fourier expansion (25). Thus,

$$u^{(m)}(x) \approx \sum_{k=-N/2+1}^{N/2} (ik)^m \widehat{u}_k \, e^{ikx} \tag{28}$$

which implies that the expansion coefficients of $u^{(m)}(x)$, $\widehat{u}_k^{(m)}$ are approximated by

$$\widehat{u}_k^{(m)} \approx (ik)^m \widehat{u}_k \tag{29}$$

It follows from the above expressions that the accuracy of the spectral differentiation depends, on one hand, on the rate of decay of the coefficients of $u(x)$, which determines how well represented is $u(x)$ by the truncated expansion, and also on the order of differentiation, which decreases the smoothness of the integrands in the transform integrals, thus increasing the quadrature error.

According to (28), the derivatives of $u(x)$ can be approximated by computing its Fourier coefficients, multiplying them by powers of ik, and then transforming back to physical space. Using the FFT the number of operations behaves like $O(Nlog(N))$. The following function, `spdiff1D(m,N)` approximates the m-th derivative of the function (27) for several p's, using N modes. With the commands `spdiff1D(1,4:4:256)` and `spdiff1D(2,4:4:256)` we may evaluate the convergence of the spectral differentiation for the first and second order derivatives, producing the plots in figure 3. It is apparent that the performance of the spectral differentiation for each particular function resembles the decay of its expansion coefficients (figure 1).

```
function spdiff1D(m,N)

syms x p
a= (1-p)/(1+p);
b= (1-p*p)/(4*p);
u= (1-p^2)./( (1+p^2) - 2*p*cos(x) );
u= (u-a)*b;
du= diff(u,m);

Fu = inline(vectorize(simplify(u )));
Fdu= inline(vectorize(simplify(du)));

i= sqrt(-1);
ps= [0.1 0.5 0.8];
for ip= 1:length(ps);
    for iN=1:length(N);
        h= 2*pi/N(iN);
        x= -pi:h:pi;x= x(1:N(iN))';
        k= [0:N(iN)/2-1 -N(iN)/2:(-1)]';

        u= Fu(ps(ip),x);
        duSP= real(ifft((i*k).^m.*fft(u)));
        duex= Fdu(ps(ip),x);
        error(iN)= (1/max(abs(duex)))*sqrt(sum( (duSP-duex).^2 )/N(iN));
    end;
    figure(1);loglog(N,error,'k','marker','o','markersize',4);
    axis square;hold on;
end;
xlabel('N','fontsize',18);ylabel('error','fontsize',18);
set(gca,'fontsize',14);
```

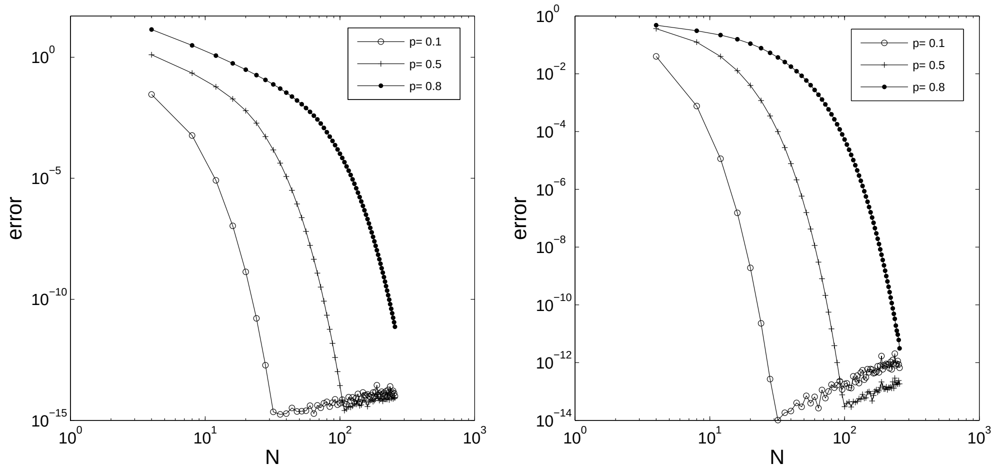


Figure 3. Convergence of the first- (left) and second-order (right) spectral derivatives for various smooth functions.

2.2. Multiple dimensions. One-dimensional spectra

The approximation framework is analogous in multiple dimensions. In 2D the Fourier expansion reads

$$u(x,y) = \sum_{k_x=-\infty}^{\infty} \sum_{k_y=-\infty}^{\infty} \widehat{u}_{k_x k_y} e^{i(k_x x + k_y y)} \tag{30}$$

with the expansion coefficients

$$\widehat{u}_{k_x k_y} = \left(\frac{1}{2\pi}\right)^2 \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} u(x,y) e^{-i(k_x x + k_y y)} dx dy \tag{31}$$

As in the one-dimensional case, we require *truncated* versions of the expansion (30), and a discretization of the transform to Fourier space (31). Thus, let us consider an (Nx, Ny)-term expansion of the form (30)

$$u(x,y) \approx \sum_{k_x=-N_x/2+1}^{N_x/2} \sum_{k_y=-N_y/2+1}^{N_y/2} \widehat{u}_{k_x k_y} e^{i(k_x x + k_y y)} \tag{32}$$

which implies a discretization with $N_x \times N_y$ *modes* in Fourier space, and a discretization in physical space using $N_x \times N_y$ grid points, and their associated grid function $\{u_{lm}, l = 1, \ldots, N_x, m = 1, \ldots, N_y\}$. The continuous transform (31) may be turned into a discrete transform through the use of the nodes as quadrature points, as

$$\widehat{u}_{k_x k_y} \approx \frac{\Delta x \Delta y}{4\pi^2} \sum_{l=1}^{N_x} \sum_{m=1}^{N_y} u_{lm} e^{-i(k_x x_{lm} + k_y y_{lm})} \tag{33}$$

Consider the 2D version of the family of functions $f(p,x)$ (equation (27)) as the tensor-product

$$u(p,x,y) = f(p,x)f(p,y) \qquad (x,y) \in [-\pi,\pi] \times [-\pi,\pi] \tag{34}$$

The Fourier coefficients of several members of this family of functions are computed using the code `coefs2D`. The only difference with respect to the 1D version is that now we have to make use of the two-dimensional fast Fourier transform (the function `fft2`). Figure 4 shows the contours associated to the absolute value of the Fourier coefficients for various values of p.

A more convenient representation of the spectrum of a multidimensional function is through the construction of its one-dimensional spectrum. Recall that the wave number is now a vector $\boldsymbol{k} = (k_x, k_y)$. Given a function $u(x,y)$, with Fourier coefficients $\widehat{u}_{\boldsymbol{k}}$, the one-dimensional spectrum $E(|k|)$ is defined as

$$E(|k|) = \frac{1}{N_{|k|}} \sum_{|k|-\frac{1}{2}}^{|k|+\frac{1}{2}} |\widehat{u}_{\boldsymbol{k}}| \tag{35}$$

Simply put, the plane (k_x, k_y) is divided into “shells” of thickness 1, and the spectrum $E(|k)$ associated to each of those shells $|k|$ is the mean of the absolute values of the Fourier coefficients lying inside that shell; i.e. those coefficients associated to modes (k_x, k_y) such that

$$|k| - \frac{1}{2} \le \sqrt{k_x^2 + k_y^2} \le |k| + \frac{1}{2} \tag{36}$$

The code `spectra1D.m`, which uses the function `[ks,sp]= spectrum(u,kmax)`, generates the 1D spectra associated to the functions in figure 4. The outcome is depicted in figure 5.

Spectral differentiation in multiple dimensions is completely analogous to the 1D case. For example, the m-th derivative of $u(x,y)$ with respect to x can be approximated by

$$u^{(m)}(x,y) \approx \sum_{k_x=-N_x/2+1}^{N_x/2} \sum_{k_y=-N_y/2+1}^{N_y/2} (ik_x)^m \widehat{u}_{k_x k_y} e^{i(k_x x + k_y y)} \tag{37}$$

Very similar expressions can be used for the derivatives with respect to y, as well as for mixed derivatives. The function `spdiff2D` analyzes the convergence of the spectral first and second order derivatives with respect to x and y. The functions being tested are the same ones as in the previous examples. The plots in figure 6 were generated using the command `spdiff2D(4:4:256)`. The only basic difference with respect to the 1D case is the use of `fft2` and `ifft2`

2.3. Solving the Laplacian using FFT’s

The elliptic problem for the stream function will be solved in Fourier space, making use of Fast Fourier Transforms. Consider for example, the boundary-value problem

$$\Delta u(x,y) = S(x,y) \tag{38}$$

with periodic boundary conditions in $[-\pi,\pi]$ and $S(x,y)$ periodic in $[-\pi,\pi]$. In Fourier space, the above equation reduces to

$$-(k_x^2 + k_y^2)\widehat{u} = \widehat{S} \tag{39}$$

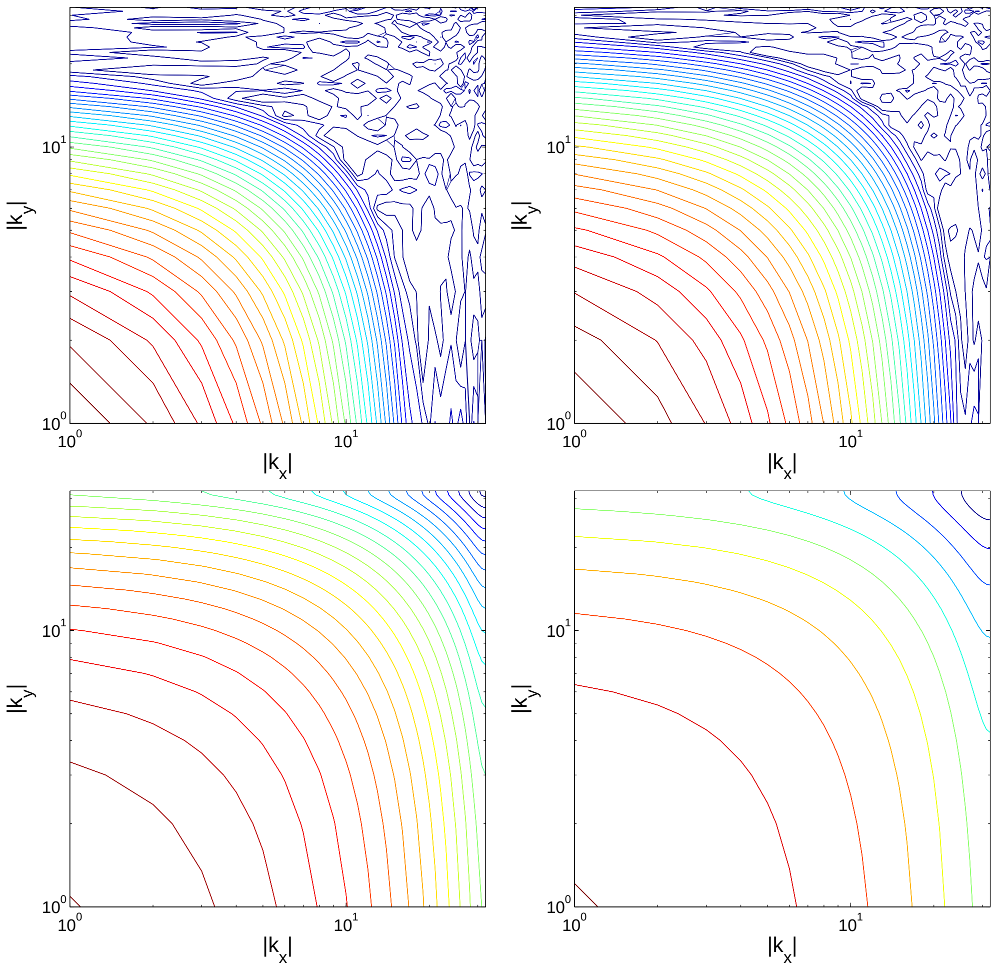


Figure 4. Contours of the absolute value of the Fourier coefficients of various 2D functions. Top, $p = 0.1$ (left) and $p = 0.2$ (right); bottom, $p = 0.6$ (left) and $p = 0.8$ (right).

and therefore the Fourier transform of $u(x, y)$, $\widehat{u}(k_x, k_y)$, is given in terms of the transform of the source term, as

$$\widehat{u} = -\frac{\widehat{S}}{k_x^2 + k_y^2} \tag{40}$$

Note that, when transforming back to physical space, $u(x, y)$ is determined up to its mean value, a fact that may be expected from the singularity of the right hand side of (40) for $k_x = k_y = 0$.

In terms of numerical discretization, the procedure goes as follows:

- Compute the Fourier coefficients of the source term, a periodic grid function $\{S_{lm}\}$, using `fft2`.
- Divide these coefficients by $-(k_x^2 + k_y^2)$, removing the singularity at the origin.

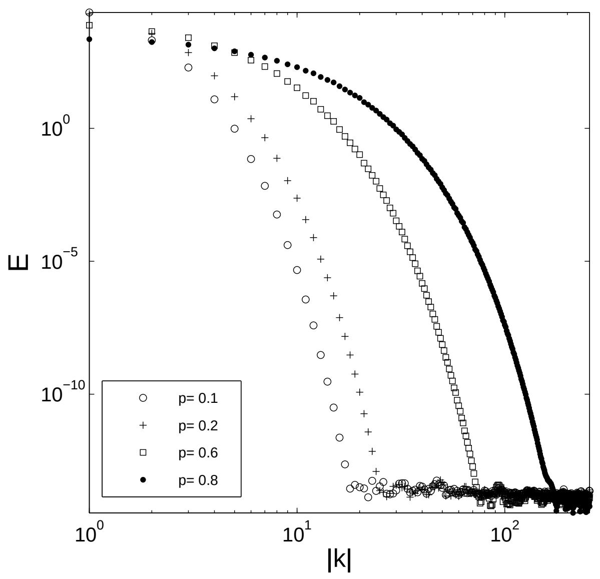


Figure 5. 1D spectra of various two-dimensional functions.

- Transform back to physical space, using `ifft2`.
- Add a constant to $u(x, y)$ in order to obtain the correct mean value.

The code `PoissonFFT2D` is an example of the implementation of these steps. A typical result with $N = 64$ modes is shown in figure 7

3. APPLICATION TO MISCIBLE FLOW THROUGH POROUS MEDIA

The code `misc2D` solves the problem (16) in a doubly periodic domain, using the Fourier approximations described in the previous sections. A typical concentration field is shown in figure 9. In addition to plotting the concentration or velocity fields, there are other interesting quantities that may be used to characterize the flow (you will have to add them to the supplied code). In particular, we will analyze the *scalar dissipation rate*, ϵ_c, defined as

$$\epsilon_c = \frac{1}{Pe}\nabla c \cdot \nabla c \qquad (41)$$

As defined above, ϵ_c is a function of space, but we will concentrate on its mean value over the domain. You will have to compute 1D spectra, rms values and probability density functions (pdf's, which can be generated in Matlab using histograms with `hist`).

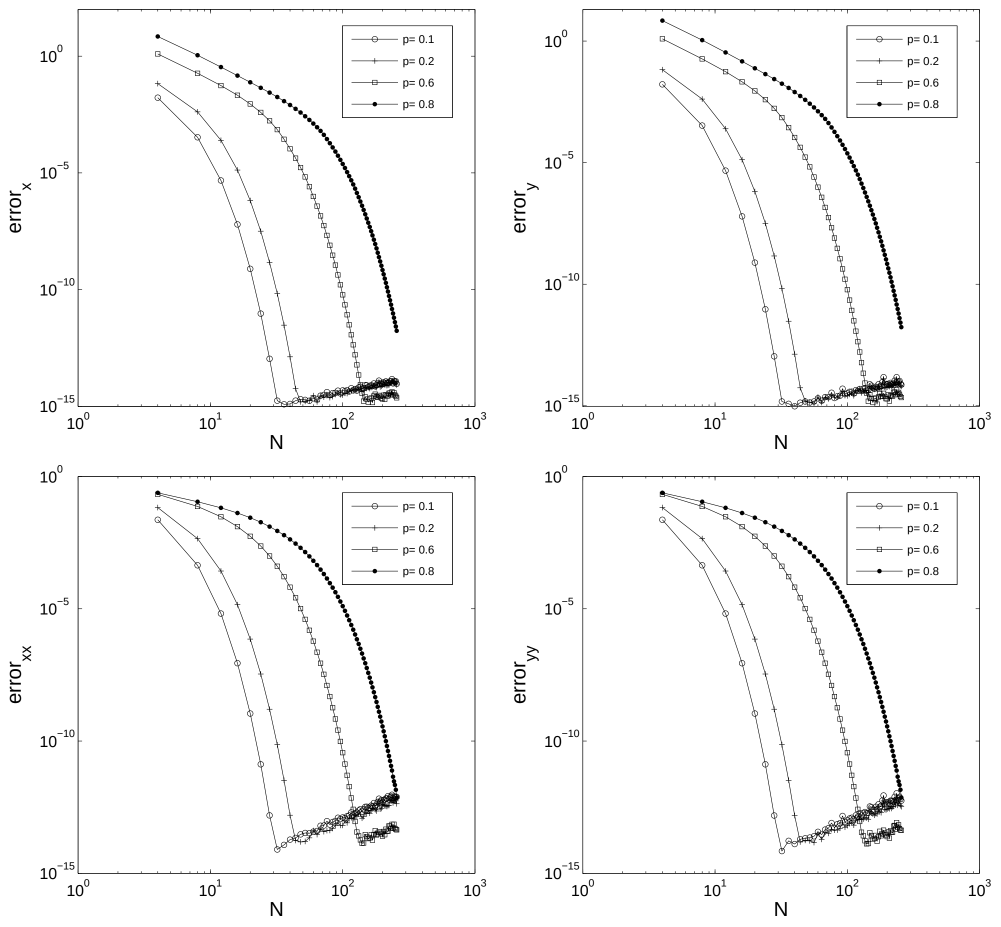


Figure 6. Convergence of the first- (top) and second-order (bottom) spectral derivatives for several two-dimensional smooth functions.

```
x= -pi:h:pi;x= x(1:N)';
[xx,yy]= meshgrid(x,x);
k= [0:N/2-1 -N/2:(-1)]';
[kx,ky]= meshgrid(k,k);
k2i= -(kx.^2 + ky.^2);k2i(1,1)= 1;k2i= k2i.^-1;

Shat= fft2(S);
u= real(ifft2(k2i.*Shat));
u= u-u(1,1);
```

REFERENCES

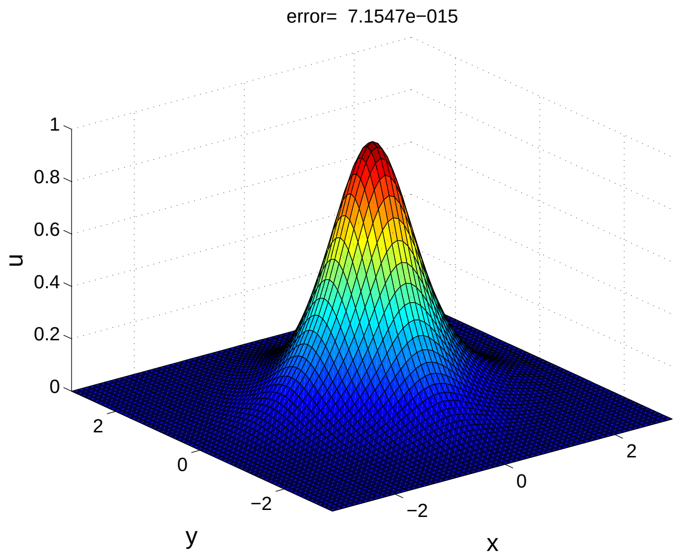


Figure 7. Solving Laplacians with the FFT.

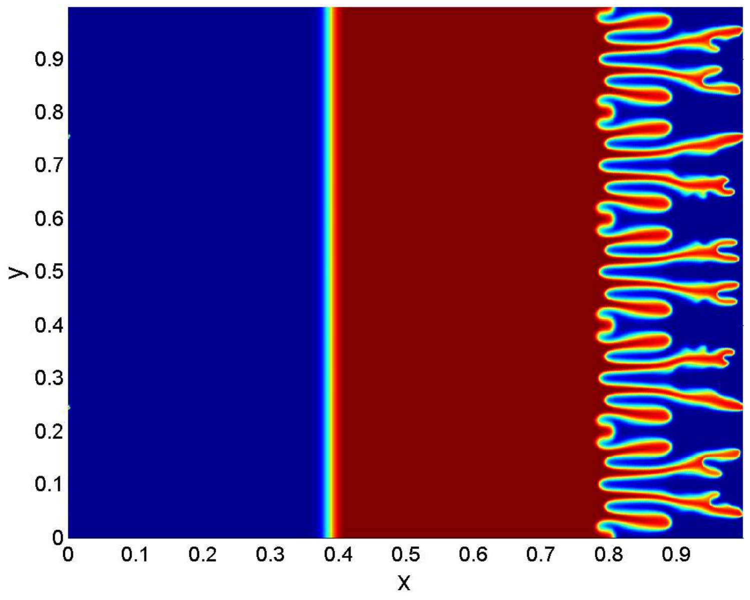
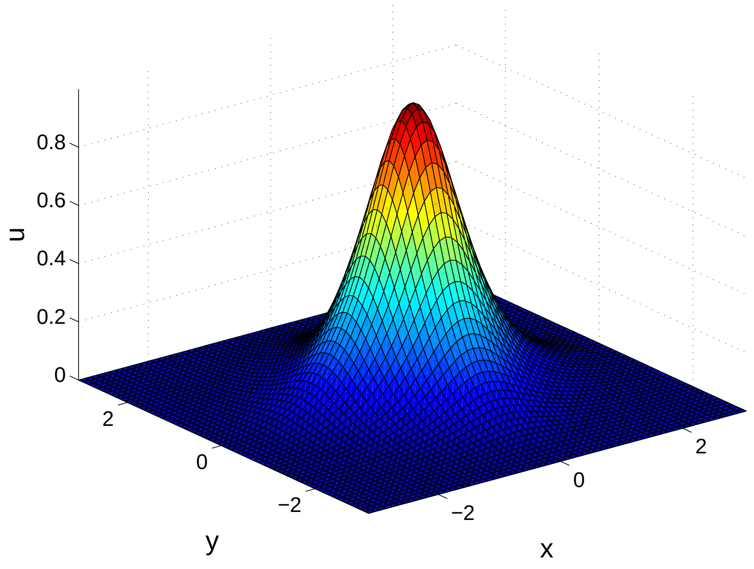


Figure 8. Sample concentration field computed with $misc2D$.



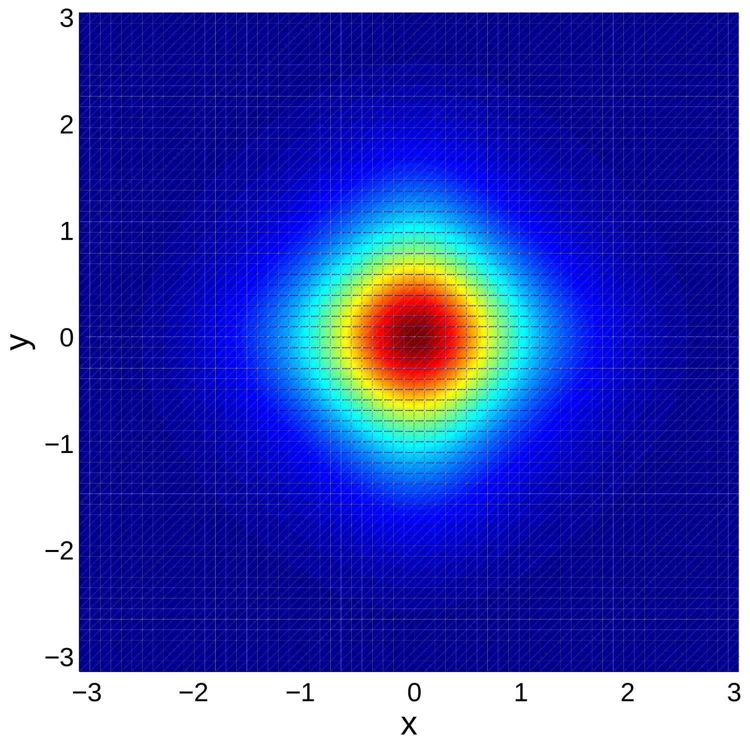


Figure 9.

1. L.N. Trefethen. Spectral methods in MATLAB. SIAM (2001)
2. J.P. Boyd. Chebyshev and Fourier spectral methods. Dover (1999)