Computational homogenisation of a 2D diffusion or waves with high contrast inclusion

A.J. Roberts

October 23, 2023

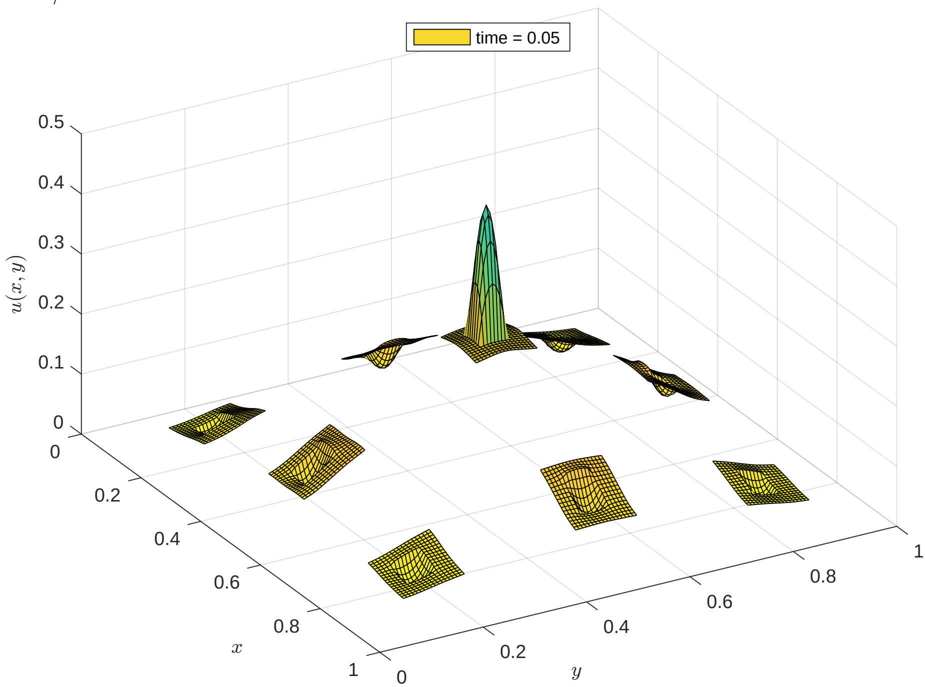
The issues raised by Elise Fressart and Barbara Verfurth 2303.15151 are the homogenisation-like modelling of wave propagation through material with microscale high-contrast 'elasticity'. Here I address the analogous problem of diffusion in high-contrast material as the issues are much the same, the discussion is more rigorous, and cognate results for waves are deduced by taking the square-root of the eigenvalues in order to get frequencies of the waves.

One can change the parameters of this code, but as is it solve diffusion in a 2D domain of $[0, 1]^2$ with macroscale periodic boundary conditions. The heterogeneity period ϵ of each cell is set to 1/9. Within each cell, it resolves the sub-period structure on a 16×16 microscale grid of spacing $dx = 0.0069$ which is perfectly adequate for demonstrating the typical behaviour. As done by Fressart and Verfurth, The heterogeneity is that the diffusion (elasticity) coefficient is one outside the centre square of each cell, and a_0 inside the centre square—they reported the cases $a_0 \in \{1/2, 1/2^5, 1/2^{10}\}$.

To clearly differentiate, where possible, the difference between the desired macroscale homogenisation and the microscale sub-cell dynamics, I here invoke the patch scheme, see Figure 1. I choose to only resolve macroscale modes with wavelengths longer than 0.5 by choosing 3×3 patches in the domain. Each patch is one cell, hence of side length 1/9. Where discussed, the sub-patch dynamics are essentially the same as the sub-cell dynamics. The patches are coupled by spectral interpolation to ensure high accuracy for any macroscale modes—whatever the 'macroscale modes' might be.

Example of homogeneous diffusion/waves The example of diffusion in homogeneous material, $a_0 = 1$, illustrates the distinction that the patch scheme

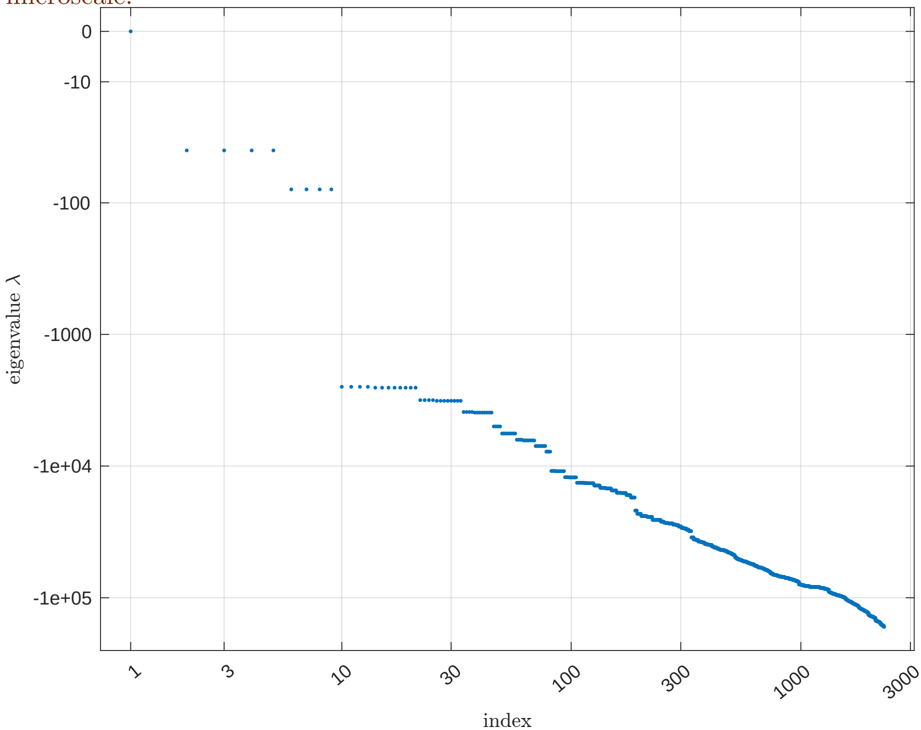
Figure 1: example of patch scheme with nine patches on the unit square simulated to time 0.05. With inclusions having low diffusivity $a_0 = 1/2^7$, the diffusion into and out of the inclusions takes quite a long time. For waves, the waves within each inclusion would bounce around inside the inclusion and only slowly leak/radiate outside.



makes between macroscale and sub-cell modes. We easily characterise the dynamics of the problem by exploring the eigenvalues. Figure 2 plots all the eigenvalues, as a function of their index on quasi-log-log axes.

- The sole eigenvalue $\lambda = 0$ represents conservation of stuff.
- The next group of four at $\lambda = -39$ represent the four macroscale modes/waves with wavenumber $(0, \pm 2\pi)$ or $(\pm 2\pi, 0)$.
- The next group of four at $\lambda = -79$ represent the four macroscale modes/waves with wavenumber $(\pm 2\pi, \pm 2\pi)$.
- The remaining eigenvalues $\lambda < -2500$ represent high-wavenumber, micro-

Figure 2: eigenvalues of the homogeneous diffusion ($a_0 = 1$). It shows a spectral gap from -79 to -2501 separating the macroscale of interest from the sub-cell microscale.

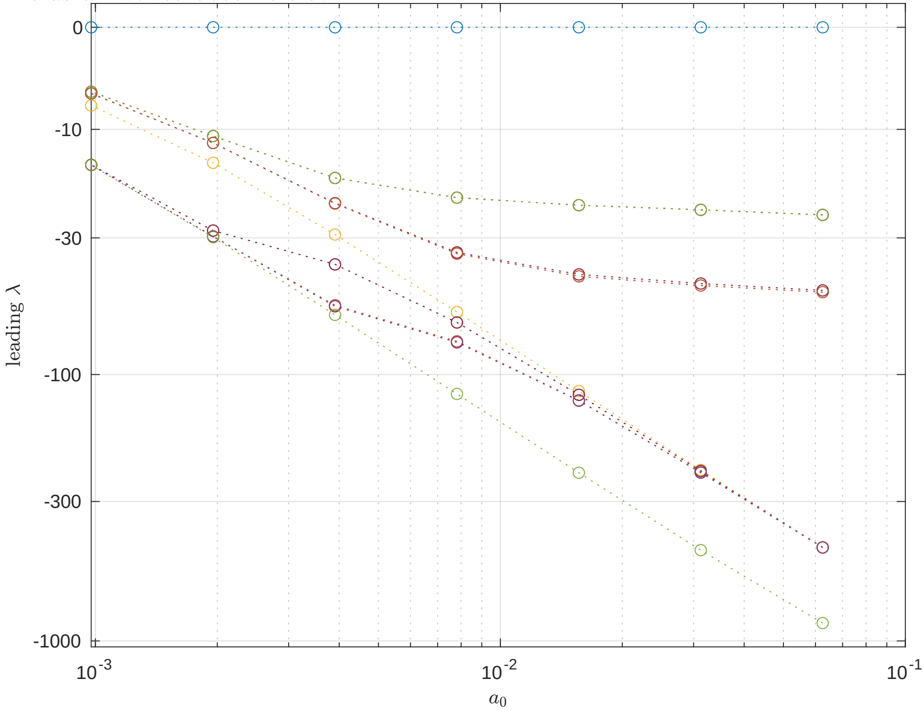


scale, sub-cell modes separated by a spectral gap of ratio 31.

In wave problems, the spectral gap would be between slow, macroscale waves of frequencies < 9, and fast, microscale, sub-cell, waves of frequencies > 50.

High contrast erodes the spectral gap Decreasing the diffusivity inside the inclusion down to $a_0 = 2^{-10}$ changes the problem to one of high-contrast. Figure 3 plots the leading eigenvalues as a function of a_0. For such very small a_0, all the sub-inclusion modes decay slowly so the become long-lasting modes, and so should be considered as part of the 'homogenised' modelling—unless one can guarantee from initial conditions (or otherwise) that they do not arise. For waves, for such very small a_0, all the sub-inclusion modes become low frequency waves

Figure 3: leading 19 eigenvalues of heterogeneous diffusion as function of the inclusion's diffusivity a_0. The spectral gap for large $a_0 \approx 0.1$ closes as a_0 decreases through 0.01 as sub-inclusion modes become long-lasting—equivalently as sub-inclusion waves become slow.



and so similarly should be considered as part of the 'homogenised' modelling—unless the initial conditions (or otherwise) ensure that they do not arise.

1 `hiContrastDiff2`: computational homogenisation of a 2D diffusion with high contrast inclusion

First set heterogeneous diffusivities constant in each of inclusion and exterior.

```
a0 = 1/2^7
mPeriod = 16
xi=(0.5:mPeriod)/mPeriod;
incl = (abs(xi'-1/2)<1/4)&(abs(xi-1/2)<1/4);
cHetr = incl*a0+(~incl)*1;
```

Configure the patch scheme with some arbitrary choices of domain, patches, size ratios. Use macroscale periodic and spectral interpolation. In 2D we get only real eigenvalues by using edgy interpolation.

```
edgyInt = true;
nPatch = 3
nSubP = mPeriod+2
dx = 1/(mPeriod*nPatch) % this is for full domain
dx = dx/3 % use smaller periodicity separated by gaps
configPatches2(@heteroDiff2,[0 1],'periodic',nPatch ...
    ,0,dx,nSubP ,'EdgyInt',edgyInt ,'hetCoeffs',cHetr );
```

Simulate Set initial conditions of a simulation (although what is FVs v0?).

```
global patches
sigma = 0.1
u0 = exp( -(patches.x-0.5).^2/sigma^2-(patches.y-0.5).^2/sigma^2 );
```

Integrate using standard integrators, unevenly spaced in time to better display transients.

```
    [ts,us] = ode23(@patchSys2, 0.05*linspace(0,1).^2, u0(:));
```

Plot the solution as an animation over time.

```
disp('plot animation of solution field')
figure(1), clf, colormap(flipud(parula))
```

Get spatial coordinates and pad them with NaNs to separate patches.

```
x = squeeze(patches.x); y = squeeze(patches.y);
x(end+1,:)=nan;  y(end+1,:)=nan; % pad with nans
```

For every time step draw the surface and pause for a short display.

```
for i = 1:length(ts)
```

Get the row vector of data, form into the 6D array via the interpolation to the edges, then pad with Nans between patches, and reshape to suit the surf function.

```
  u = squeeze( mean( patchEdgeInt2(us(i,:)) ,4));
  u(end+1,:,:,:)=nan; u(:,end+1,:,:)=nan;
  u = reshape(permute(u,[1 3 2 4]), [numel(x) numel(y)]);
```

If the initial time then draw the surface with labels, otherwise just update the surface data.

```
  if i==1
       hsurf = surf(x(:),y(:),u'); view(60,40)
       xlim([0 1]), ylim([0 1]),   caxis([0 1])
       xlabel('$x$'), ylabel('$y$'), zlabel('$u(x,y)$')
  else set(hsurf,'ZData', u');
  end
  legend(['time = ' num2str(ts(i),2)],'Location','north')
  pause(0.05)
```

finish the animation loop and if-plot.

```
end%for over time
```

1.1 Compute Jacobian and its spectrum

Let's explore the dynamics via the Jacobian. Find which elements of the 6D array are interior micro-grid points and hence correspond to dynamical variables.

```
    u0 = zeros([nSubP,nSubP,1,1,nPatch,nPatch]);
    u0([1 end],:,:) = nan;
    u0(:,[1 end],:) = nan;
    i = find(~isnan(u0));
```

Construct the Jacobian of the scheme as the matrix of the linear transformation, obtained by transforming the standard unit vectors.

```
    Jac = nan(length(i));
    sizeJacobian = size(Jac)
    for j = 1:length(i)
      u = u0(:)+(i(j)==(1:numel(u0))');
      tmp = patchSys2(0,u);
```

```
      Jac(:,j) = tmp(i);
    end
```

Test for symmetry, with error if we know it should be symmetric.

```
    notSymmetric=norm(Jac-Jac')
    assert(notSymmetric<1e-7,'failed symmetry')
```

Find all the eigenvalues (as `eigs` is unreliable).

```
    [evecs,evals] = eig((Jac+Jac')/2,'vector');
```

Sort eigenvalues on their real-part with most positive first, and most negative last. List leading, and plot all.

```
    [~,k] = sort(-real(evals));
    evals=evals(k); evecs=evecs(:,k);
    leadingEvals=evals(1:2*nPatch^2+1)
    figure(2),clf
    plot(evals,'.')
    xlabel('index'),ylabel('eigenvalue $\lambda$')
    quasiLogAxes(1,10)
```

1.2 `heteroDiff2()`: heterogeneous diffusion

This function codes the lattice heterogeneous diffusion inside the patches. For 6D input arrays `u`, `x`, and `y` (via edge-value interpolation of `patchSys2`), computes the time derivative at each point in the interior of a patch, output in `ut`. The two 2D array of diffusivities, c^x_{ij} and c^y_{ij}, have previously been stored in `patches.cs` (3D).

```
function ut = heteroDiff2(t,u,patches)
  dx = diff(patches.x(2:3));  % x space step
  dy = diff(patches.y(2:3));  % y space step
  ix = 2:size(u,1)-1; % x interior points in a patch
  iy = 2:size(u,2)-1; % y interior points in a patch
  ut = nan+u;         % preallocate output array
  ut(ix,iy,:,:,:,:) ...
  = diff(patches.cs(:,iy,:).*diff(u(:,iy,:,:,:,:),1),1)/dx^2 ...
   +diff(patches.cs(ix,:,:).*diff(u(ix,:,:,:,:,:),1,2),1,2)/dy^2;
end% function
```