`nearhomoDiff1`: computational homogenisation of a 1D heterogeneous diffusion with nearly correct period

AJR

September 2, 2023

Suppose the spatial microscale lattice is at points x_i, with constant spacing dx. With dependent variables $u_i(t)$, simulate the microscale lattice diffusion system

$$\frac{\partial u_i}{\partial t} = \frac{1}{dx^2}\delta[c_{i-1/2}\delta u_i], \tag{1}$$

in terms of the centred difference operator δ. The system has a microscale heterogeneity via the coefficients $c_{i+1/2}$ which we assume to have some periodicity.

0.1 Code various numbers of patches over domain

Establish system of length 2π and a heterogeneity so the eigenvalues should be close to $0, -1, -4, -9, \ldots$. Explore variety of number of patches.

```
%}
Xlim = [-pi pi]
lMax = 4
nPatches = 7*3.^(0:lMax-1)
%{
```

Set up microgrid parameters. CDE used cell size $\eta/\epsilon \in [1, 50]$, and at least 4 096 points, so here with, say, 189 cells that is over 21 points per cell.

```
%}
mPerPatch = 12
eta = diff(Xlim)/nPatches(lMax)
dx = eta/mPerPatch
%{
```

Start with the periodicity equal to to cell-size, and set strength of heterogeneity (abs-value less than one). Now the micro-scale heterogeneity must be 2π-periodic so in general has to be of the following form for integer `nDetune` (positive means smaller period as in CDE, negative means larger).

```
%}
nDetune = 1
epsilon = 2*pi/(2*pi/eta+nDetune)
heteroAmp = 0.9 % 0.9 is close to CDE's (4.1a)
%{
```

Consequently, the heterogeneity

$$\begin{aligned}\cos(2\pi x/\epsilon) &= \cos\left[2\pi x/\eta + 2\pi(1/\epsilon - 1/\eta)x\right]\\ &= \cos(2\pi x/\eta)\cos(kx) - \sin(2\pi x/\eta)\sin(kx)\\ &\text{for wavenumber } k := 2\pi(1/\epsilon - 1/\eta).\end{aligned}$$

So the discrepancy can be viewed as a modulation of precise cell-periodicity by variations of wavenumber k. That is, the discrepancy may be viewed as an example of a "functionally graded material". In the patch scheme such modulations are resolved on patches of spacing H provided their wavenumber $k < \pi/H$. That is, we require

$$\begin{aligned}&2\pi(1/\epsilon - 1/\eta) < \pi/H\\ \iff{}& \eta/\epsilon - 1 < (\eta/H)/2\\ \iff{}& \eta/\epsilon < 1 + r/2\end{aligned}$$

where $r := \eta/H$ is the patch ratio. For example, for $r = \frac{1}{3}, \frac{1}{9}$ we need $\frac{\eta}{\epsilon} < \frac{7}{6}, \frac{19}{18} \approx 1.17, 1.06$ in order to realise accuracy.

0.2 Code to create the patch schemes

Establish the global data struct `patches` for the microscale heterogeneous lattice diffusion system (1) solved on 2π-periodic domain. Use spectral interpolation for best accuracy.

```
%}
global patches
leadingEvals=[];
for nPatch = nPatches
    nSubP = mPerPatch+2;
    configPatches1(@heteroDiff,Xlim,'periodic',nPatch ...
        ,0,dx,nSubP,'EdgyInt',true);
%{
```

Set the microscale heterogeneity with harmonic mean one.

```
%}
xMid = ( patches.x(1:end-1,:,:,:)+patches.x(2:end,:,:,:) )/2;
patches.cs = 1./(1+ heteroAmp*cos(2*pi*xMid/epsilon) );
%xMid=squeeze(xMid)
%xMicro=squeeze(patches.x)
%cMicro = squeeze(patches.cs)
%{
```

Compute Jacobian and its spectrum Form the Jacobian matrix, linear operator, by numerical construction about a zero field. Use `i` to store the indices of the micro-grid points that are interior to the patches and hence are the system variables. The detuned periodicities are non-symmetric so no point checking for symmetry.

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

```
  for j=1:nJac
    u0(i)=((1:nJac)==j);
    dudt=patchSys1(0,u0);
    Jac(:,j)=dudt(i);
  end
%{
```

Find the eigenvalues of the Jacobian, and list for inspection: the spectral interpolation is effectively exact for the macroscale.

The number of zero eigenvalues, `nZeroEv`, indicates the number of decoupled systems in this patch configuration.

```
%}
  if nPatch<nPatches(end)
        [evecs,evals]=eig(Jac,'vector');
        tol=1e-6; %zero evec elements with small components
        j=find(abs(real(evecs))<tol); evecs(j)=imag(evecs(j));
        j=find(abs(imag(evecs))<tol); evecs(j)=real(evecs(j));
        % sort on the number of zero crossings mod by eval
        n0x = sum(abs(diff(sign(real(evecs([1:end 1],:)))))) ...
             +sum(abs(diff(sign(imag(evecs([1:end 1],:))))));
        [n0x,j]=sort( n0x -asinh(real(evals'))/100 );
        evals=evals(j);
  else nonSymmetric=norm(Jac-Jac')
        assert(nonSymmetric<1e-6,'failed symmetry')
        evals=eigs(sparse(Jac+Jac')/2,nPatches(1),-0.3);
        evals=sort(evals,'descend');
  end
%  nZeroEv=sum(abs(evals)<1e-5)
  leadingEvals=[leadingEvals evals(1:nPatches(1))];
%{
```

End of the for-loop over the number of patches.

```
%}
end%for nPatch
```

```
nPatches = nPatches
maxImagLeadingEvals = max(abs(imag(leadingEvals(:))))
reLeadingEvals = real(leadingEvals)
log10errLeadingEvals = log10(abs( leadingEvals-leadingEvals(:,lMax)
eta2epsilonRatio = eta/epsilon
disp('****log10 error leading evals')
disp(num2str(log10errLeadingEvals,2))
%{
```

End of the main script.

0.3 `heteroDiff()`: heterogeneous diffusion

This function codes the lattice heterogeneous diffusion inside the patches. For 2D input arrays `u` and `x` (via edge-value interpolation of `patchSys1`, **??**), computes the time derivative **??** at each point in the interior of a patch, output in `ut`. The array of diffusivities c_i have previously been stored in struct `patches.cs`.

```
%}
function ut = heteroDiff(t,u,patches)
  dx = diff(patches.x(2:3));   % space step
  i = 2:size(u,1)-1;   % interior points in a patch
  ut = nan+u;          % preallocate output array
  ut(i,:,:,:) = diff(patches.cs(:,1,:,:).*diff(u))/dx^2 ...
                -0*u(i,:,:,:);
end% function
%{
```