Holistic discretisation of wave-like PDEs, II

Tony Roberts Meng Cao

December 15, 2011

Try to develop good numerics of wave-like PDEs using a staggered element approach. For 'small' parameter ν, the PDEs for fields $h(x,t)$ and $u(x,t)$ are among

$$\frac{\partial h}{\partial t} = -\frac{\partial u}{\partial x}\,,$$
$$\frac{\partial u}{\partial t} = -\frac{\partial h}{\partial x} - \nu u + \nu\frac{\partial^2 u}{\partial x^2} - \nu u\frac{\partial u}{\partial x}\,.$$

To be solved on elements centred on X_j, $X_j = jD$ say, with some coupling condition. The difference here is that we let the elements overlap so the jth element is the interval $E_j = (X_{j-1}, X_{j+1})$.

Because of even/odd symmetry I think it is more convenient to imagine two fields, each on overlapping elements, for each physical field: in element E_j introduce $h_j(x,t)$, $h'_j(x,t)$, $u_j(x,t)$ and $u'_j(x,t)$. I aim to eventuates that even-undashed fields interact with odd-undashed fields, and vice versa, but that the two sets of fields do not interact with the other. The PDEs are then

j odd (even)

$$\frac{\partial h'_j}{\partial t} = -\frac{\partial u_j}{\partial x}\,,$$
$$\frac{\partial u_j}{\partial t} = -\frac{\partial h'_j}{\partial x} - \nu u_j\,,$$

j even (odd)

$$\frac{\partial h_j}{\partial t} = -\frac{\partial u'_j}{\partial x}\,,$$
$$\frac{\partial u'_j}{\partial t} = -\frac{\partial h_j}{\partial x} - \nu u'_j\,.$$

Here I propose the coupling condition on the h_j fields, j even (odd), of

$$h_{j\pm1}(X_j) = h'_j(X_j) \pm (1-\gamma)\tfrac{1}{2}\big[h_{j+1}(X_{j+2}) - h_{j-1}(X_{j-2})\big],$$

and correspondingly couple the u_j fields, j odd (even), with

$$u_{j\pm1}(X_j) = u'_j(X_j) \pm (1-\gamma)\tfrac{1}{2}\big[u_{j+1}(X_{j+2}) - u_{j-1}(X_{j-2})\big].$$

Then define the amplitudes to be

$$H_j = h'_j(X_j) \quad \text{and} \quad U_j = u'_j(X_j),$$

respectively for j even and odd (odd and even). Be careful with the dashes.

1 Computer algebra constructs the model

Improve printing

```
on div; off allfac; on revpri;
linelength(70)$ factor dd,df;
```

Avoid slow integration with specific operator Introduce the sign function to handle the derivative discontinuities across the centre of each element. Define the integral operator to handle polynomials with sign functions, both indefinite ($\int_0^\xi d\xi$) and definite to $\xi = \mathtt{q} = \pm1$ ($\int_0^q d\xi$).

```
let df(sign(~x),~y)=>0;
operator intx; linear intx;
let { intx(xi^~~p,xi)=>xi^(p+1)/(p+1)
    , intx(1,xi)=>xi
    , intx(sign(xi)*xi^~~p,xi)=>sign(xi)*xi^(p+1)/(p+1)
    , intx(sign(xi),xi)=>sign(xi)*xi
    , intx(xi^~~p,xi,~q)=>q^(p+1)/(p+1)
```

```
10      , intx(1,xi,~q)=>q
11      , intx(sign(xi)*xi^~~p,xi,~q)=>sign(q)*q^(p+1)/(p+1)
12      , intx(sign(xi),xi,~q)=>sign(q)*q
13      };
```

Do I need this? Do not seem to need the sign function.

Introduce subgrid variable Introduced above is the subgrid variable $\xi = (x - X_j)/D$, $|\xi| < 1$, in which the fields are described.

```
14 depend xi,x;  let df(xi,x)=>1/dd;
```

Define evolving amplitudes Amplitudes are as $U_j(t) = u'_j(X_j, t)$ and $H_j(t) = h'_j(X_j, t)$, The difference here is that we take, say, even j to be the u-elements and odd j to be the h-elements. Actually it should not matter which way around, or even if you regard the modelling as being of two disjoint systems (one one way and one the other). The amplitudes depend upon time according to some approximation stored in `gh` and `gu`.

```
15 operator hh; operator uu;
16 depend hh,t; depend uu,t;
17 let { df(hh(~k),t)=>sub(j=k,gh)
18      , df(uu(~k),t)=>sub(j=k,gu)
19      };
```

But solvability condition is coupled Now the evolution equations are coupled together. By some symmetry we decouple the equations using this operator `ginv`. However, I expect that some problems will not decouple (look for non-cancelling pollution by `ginv` operators). In which case we have to accept that the DEs for the amplitudes are *implicit* DEs using the following operator. Let's define $\mathcal{G} = E + E^{-1}$ so that $\mathcal{G}F_j = F_{j+1} + F_{j-1}$. Take $\mathcal{G}^{-1}$ of this equation to deduce $\mathcal{G}^{-1}F_{j\pm1} = F_j - \mathcal{G}^{-1}F_{j\mp1}$, and change subscripts,

$j \mapsto k \mp 1$, to deduce $\mathcal{G}^{-1}F_k = F_{k\mp1} - \mathcal{G}^{-1}F_{k\mp2}$. That is, we change an inverse of $\mathcal{G}$ to one with subscript closer to $k = j$, or otherwise if we desire. Have here coded some quadratic transformations so we can resolve quadratic terms in the model, but I guess we also might want cubic.

The following causes a warning that `~a` and `~b` are declared operator, which is fine, but I cannot predefine them as operators so cannot avoid the warning.

```
operator ginv; linear ginv;
let { df(ginv(~a,t),t)=>ginv(df(a,t),t)
    , ginv(~a(j+~k),t)=>a(j+k-1)-ginv(a(j+k-2),t) when k>1
    , ginv(~a(j+~k),t)=>a(j+k+1)-ginv(a(j+k+2),t) when k<0
    , ginv(~a(j+~k)^2,t)=>a(j+k-1)^2-ginv(a(j+k-2)^2,t) when k
    , ginv(~a(j+~k)^2,t)=>a(j+k+1)^2-ginv(a(j+k+2)^2,t) when k
    , ginv(~a(j+~~k)*~b(j+~~l),t)=> a(j+k-1)*b(j+l-1)
      -ginv(a(j+k-2)*b(j+l-2),t) when k+l>2
    , ginv(~a(j+~~k)*~b(j+~~l),t)=> a(j+k+1)*b(j+l+1)
      -ginv(a(j+k+2)*b(j+l+2),t) when k+l<-1
    };
```

Start with linear approximation The linear approximation is the usual piecewise constant fields in each element. Except that the dashed fields are (surprisingly sensible) averages of the surrounding elements.

```
hj:=hh(j); hdj:=(hh(j+1)+hh(j-1))/2;
uj:=uu(j); udj:=(uu(j+1)+uu(j-1))/2;
gh:=gu:=0;
let gam^2=>0; factor gam;
gamma:=gam;
let nu^2=>0; factor nu;
```

Iterate to a slow manifold Iterate to seek a solution, terminating only when residuals are zero to specified order.

```
37 for it:=1:4 do begin
38 write "ITERATION = ",it;
```

Choose this order of updating fields from residuals due to the pattern of communication.

First do the equations for the evolution of the dashed fields. j even

```
39 resud:=df(udj,t)+df(hj,x)+nu*udj;
40 write lengthresud:=length(resud);
41 reshb:=sub({j=j+1,xi=-1},hj)-sub({j=j-1,xi=+1},hj)
42     -(1-gamma)*(sub({j=j+1,xi=+1},hj)-sub({j=j-1,xi=-1},hj));
43 write lengthreshb:=length(reshb);
44 gu:=gu+(gud:=-ginv(ginv(reshb/dd,t)
45     +intx(resud,xi,1)-intx(resud,xi,-1),t));
46 hj:=hj-dd*intx(resud+sub(j=j-1,gud)/2+sub(j=j+1,gud)/2,xi);
```

j odd

```
47 reshd:=df(hdj,t)+df(uj,x)+nu*hdj;
48 write lengthreshd:=length(reshd);
49 resub:=sub({j=j+1,xi=-1},uj)-sub({j=j-1,xi=+1},uj)
50     -(1-gamma)*(sub({j=j+1,xi=+1},uj)-sub({j=j-1,xi=-1},uj));
51 write lengthresub:=length(resub);
52 gh:=gh+(ghd:=-ginv(ginv(resub/dd,t)
53     +intx(reshd,xi,1)-intx(reshd,xi,-1),t));
54 uj:=uj-dd*intx(reshd+sub(j=j-1,ghd)/2+sub(j=j+1,ghd)/2,xi);
```

Second do the equations for the evolution of the undashed fields, to get spatial structure of dashed fields. j even

```
55 resh:=df(hj,t)+df(udj,x);
56 write lengthresh:=length(resh);
57 resua:=-sub(xi=0,udj)
```

```
58      +sub({j=j+1,xi=-1},uj)/2+sub({j=j-1,xi=+1},uj)/2;
59  write lengthresua:=length(resua);
60  udj:=udj+resua-dd*int(resh,xi);
```

j odd

```
61  resu:=df(uj,t)+df(hdj,x)+nu*uj;
62  write lengthresu:=length(resu);
63  resha:=-sub(xi=0,hdj)
64      +sub({j=j+1,xi=-1},hj)/2+sub({j=j-1,xi=+1},hj)/2;
65  write lengthresha:=length(resha);
66  hdj:=hdj+resha-dd*intx(resu,xi);
```

Terminate the loop Exit the loop if all residuals are zero.

```
67    if {resh,reshd,resha,reshb,resu,resud,resua,resub}
68    ={0,0,0,0,0,0,0,0} then write it:=it+100000;
69    showtime;
70  end;
```