Chebop/linop roadmap

Friday, December 04, 2009
3:03 PM

The current state of chebop/linop is still a transitional one. Here is how I see things progressing.

linop is a subclass of chebop, because a linear operator "is a" operator. Thus, isa(L,'chebop') and isa(L,'linop') are both true for linop L.

Currently, linops maintain their own functional expressions via the oparray class. Since chebops maintain functional expressions anyway, this part of linops should go away. (Composition of quasimatrix-style functions may end up being significantly more efficient than the oparray system.)

Chebops have boundary conditions, so linops should avoid duplicating this feature. However, the BC of a linop is always linear. Thus is it conceptually (as well as algorithmically) significant to distingish linear chebops as linops. I propose that one use the linop constructor to designate a linop:

chebop(d,@(u) diff(u)) will not be recognized as linear
linop(d,@(u) diff(u)) will be known as linear, and take a Jacobian to get diff(d)
diff(d) will call linop(d,@(u) diff(u),[],[],@(n) diffmat(n))

A big lingering issue is the performance on systems. Part of this may be the poor way oparrays work. That will be gone, so perhaps things will improve. Right now, horzcat and vertcat are done in "lazy" fashion, which may be inefficient as it's done over and over for the leaves of a large expression tree. Or it might be that the anon class will help, or that new ideas are needed.

chebop properties	linop properties
domain	varmat
op	difforder
all bc fields	blocksize?
ID number	
guess	
optype, if needed	