fish.gex - GrADS Extension Library for Calculating Streamfunction/Velocity Potential

NAME

fish.gex - GrADS Extension Library for Calculating Streamfunction/Velocity Potential

SYNOPSIS

GrADS Functions:

fish(*UEXPR,VEXPR[,MBDCND]*) - Poisson Solver

fish_psi(*UEXPR,VEXPR[,MBDCND]*) - Computes Stream Function

fish_chi(*UEXPR,VEXPR[,MBDCND]*) - Computes Velocity Potential

fish_vor(*UEXPR,VEXPR[,MBDCND]*) - Computes Relative Vorticity

fish_div(*UEXPR,VEXPR[,MBDCND]*) - Computes Divergence

DESCRIPTION

This library provides GrADS extensions (*gex*) with functions for computation of streamfunction and velocity potential from zonal and meridional wind components:

```
laplacian(psi) = vorticity (1)
laplacian(chi) = divergence (2)
```

where `psi` is the streamfunction and `chi` is the velocity potential. (See Wikipedia links below for more information on streamfunction/velocity potential.) The vorticity and divergence computation relies on functions **madvu** and **madvv** provided in the OpenGrADS extension *Libbjt* by B.-J. Tsuang. The Poisson equations (1)-(2) above are solved using the classic `Fishpak` Fortran library. Documentation for `FISHPAK` is given in:

Swarztrauber, P. and R. Sweet, 1975: Efficient Fortran Subprograms for the Solution of Elliptic Partial Differential Equations. *Technical Note TN/IA-109*. National Center for Atmospheric Research, Boulder, Colorado 80307.

All functions provided require a global, uniform lat/lon grid.

EXAMPLES

For the following examples it is suggested that you open the following OPeNDAP dataset with a GFS forecast. Fire up `gradsdods` and then

```
ga-> sdfopen http://nomad2.ncep.noaa.gov:9090/dods/gfs/rotating/gfs_00z
```

Computing Streamfunction from Vorticity

If you have the relative vorticity field available, say `vor`, you can easily compute streamfunction

```
ga-> psi = fish(vor)
```

Computing Streamfunction from Wind Components

The first step is to evaluate the streamfunction:

```
ga-> set lev 200
ga-> psi = fish_psi(ugrd,vgrd)
```

It is often convenient to *center* the streamfunction by subtracting its global mean:

```
ga-> psi = psi - aave(psi,global)
```

We can finally display it:

```
ga-> set gxout shaded
ga-> display psi/1e7
draw title Streamfunction
```

Computing the Rotational Wind

Although the intrinsc GrADS function `cdiff` could be used for numerically diffferentiating the streamfunction, we use here functions `mvadv/muadv` from *Libbjt* because of its handling of the boundaries:

```
ga-> one = 1 + 0 * lat
ga-> upsi = mvadv(one,psi)
ga-> vpsi = - muadv(one,psi)
```

We then plot the rotational streamlines on top of the streamfunction:

```
ga-> set gxout shaded
ga-> display psi/1e7
ga-> set gxout stream
ga-> display upsi;vpsi
ga-> draw title Streamfunction/Rotational Wind
```

Computing Velocity Potential from Divergence

If you have the divergence field available, say `div`, you can easily compute streamfunction

```
 ga-> chi = fish(div)
```

Computing Velocity Potential from Wind Components

Start by computing the velocity potential:

```
ga-> set lev 200
ga-> chi = fish_chi(ugrd,vgrd)
```

It is often convenient to *center* the velocity potential by subtracting its global mean:

```
ga-> psi = psi - aave(psi,global)
```

We can finally display it:

```
ga-> set gxout shaded
ga-> display chi/1e6
draw title Velocity
```

Calculating the Divergent Wind

Although the intrinsc GrADS function `cdiff` could be used for numerically diffferentiating the velocity potential, we use here functions `mvadv/muadv` from *Libbjt* because of its handling of the boundaries:

```
ga-> uchi = - muadv(one,chi)
ga-> vchi = - mvadv(one,chi)
```

We finally display the divergent wind as vectors on top of the velocity potential:

```
ga-> display chi/1e6
ga-> set gxout vector
ga-> set cmin 2
ga-> set cmax 20
ga-> display skip(uchi,6,6);vchi
ga-> draw title Velocity Potential/Divergent Wind
```

Computing Vorticity and Divergence

As an alternative to the intrinsic GrADS functions `hcurl/hdivg`, functions `fish_vor` and `fish_div` uses the advention functions in *Libbjt* to numerically evaluate vorticity and divergence

```
 ga-> vor = fish_vor(ugrd,vgrd)
 ga-> div = fish_div(ugrd,vgrd)
```

These functions provide a better handling of the boundaries compared to their intrinsic counterparts.

FUNCTIONS PROVIDED

fish(*UEXPR,VEXPR[,MBDCND]*) - Poisson Solver

This function uses routine `PWSSSP` in `Fishpak` to solve the poisson equation. The default parameter `MBDCND=9` solves a Helmholts equation with a very small random term to provide a "unique" solution.

UEXPR,VEXPR - required

GrADS expressions with zonal and meridional wind components

MBDCND - optional

Meridional boundary condition; the default `MBDCND=9` should work in most cases. See `FISHPAK` documentation for additional information.

```
MBDCND = 1 ! BC: solution specified at both poles
MBDCND = 5 ! BC: solution specified at TF (South Pole) and
MBDCND = 7 ! BC: solution specified at TS (North Pole) and
MBDCND = 9 ! BC: solution unspecified at both poles
```

fish_psi(*UEXPR,VEXPR[,MBDCND]*) - Computes Stream Function

This function computes vorticity as in **fish_vor** and uses **fish** to solve the Poisson equation for the streamfunction `psi`:

```
  laplacian(psi) = vorticity
```

UEXPR,VEXPR - required

GrADS expressions with zonal and meridional wind components

MBDCND - optional

Meridional boundary condition; the default `MBDCND=9` should work in most cases. See `FISHPAK` documentation for additional information.

```
        MBDCND = 1 ! BC: solution specified at both poles
        MBDCND = 5 ! BC: solution specified at TF (South Pole) and
        MBDCND = 7 ! BC: solution specified at TS (North Pole) and
        MBDCND = 9 ! BC: solution unspecified at both poles
```

fish_chi(*UEXPR,VEXPR[,MBDCND]*) - Computes Velocity Potential

This function computes divergence as in **fish_div** and uses **fish** to solve the Poisson equation for the velocity potential `chi`:

```
  laplacian(chi) = divergence
```

UEXPR,VEXPR - required

GrADS expressions with zonal and meridional wind components

MBDCND - optional

Meridional boundary condition; the default `MBDCND=9` should work in most cases. See `FISHPAK` documentation for additional information.

```
        MBDCND = 1 ! BC: solution specified at both poles
        MBDCND = 5 ! BC: solution specified at TF (South Pole) and
        MBDCND = 7 ! BC: solution specified at TS (North Pole) and
        MBDCND = 9 ! BC: solution unspecified at both poles
```

fish_vor(*UEXPR,VEXPR*) - Computes Relative Vorticity

This function computes the vorticity using the expression:

```
  vorticity = - ( madvu(v,one) - madvv(u,cosphi) / cosphi )
```

where `u` and `v` are the zonal/meridional wind components, `one` is a constant field equal to 1, and `cosphi` is the cosine of latitude. The functions **madvu** and **madvv** are provided in the OpenGrADS extension library *Libbjt*.

UEXPR,VEXPR - required

GrADS expressions with zonal and meridional wind components

fish_div(*UEXPR,VEXPR*) - Computes Divergence

This function computes the divergence using the expression:

```
   divergence = - ( madvu(u,one) + madvv(v,cosphi) / cosphi )
```

where `u` and `v` are the zonal/meridional wind components, `one` is a constant field equal to 1, and `cosphi` is the cosine of latitude. The functions **madvu** and **madvv** are provided in the OpenGrADS extension library *Libbjt*.

UEXPR,VEXPR **- required**

GrADS expressions with zonal and meridional wind components

BUGS

The function `fish` assumes a global and uniform longitude/latitude grid. If your grid is not uniform, consider using `re()` for interpolating to a uniform grid. When doing so, do not make the poles gridpoints. (The function `muadv` from *Libbjt* produces undefined values if the first and last latitudinal gridpoints are at the poles.)

Undefined values are handled in a less than idea way by the Poisson solver `fish`. If undefined values of vortivity/divergence occur at the first and last latitudinal gridpoint, the following *polar fix* is applied depending on whether these gridpoints are the poles or not:

Notice that `fish()` becomes numerically unstable for horizontal resolutions finer that 1/2 degrees or so. In such cases use the spherical harmonic based **sh_fish** given in extension *shfilt*.

Poles are gridpoints

If the first and last latitudinal gridpoints are at the poles, the zonal average of the adjascent latitudinal band is computed and this zonal averaged value is used at the pole.

Poles are not gridpoints

The value at the same longitude in the adjascent latitudinal band is used.

For interior points, undefined values are set to zero before solving the Poisson equation.

SEE ALSO

- http://opengrads.org/ - OpenGrADS Home Page
- http://opengrads.org/wiki/index.php - OpenGrADS User Defined Extensions
- http://www.iges.org/grads/ - Official GrADS Home Page
- http://en.wikipedia.org/wiki/Velocity_potential - Velocity Potential definition on Wikipedia.
- http://en.wikipedia.org/wiki/Stream_function - Stream function definition on Wikipedia.

AUTHOR

Arlindo da Silva (dasilva@opengrads.org)

COPYRIGHT

fish.gex - GrADS Extension Library for Calculating Streamfunction/Velocity Potential