Smoothing JULES code for differentiability in ADJULES

June 27, 2013

'IF', 'MAX' and 'MIN' statements within the JULES code create discontinuities in the function to be differentiated in adjoint generation. These step changes in the cost function yield zero derivatives at discontinuities. In order to solve this problem, it is possible to replace such statements with smoother alternatives. The TAF compilation option '`warn_step`' identifies lines in the code where this behaviour may occur, allowing for the identification of appropriate locations for replacement of discontinuous functions with smoother alternatives.

1 Theory

1.1 Smoothing 'IF' statements

IF statements can be smoothed as follows: Given an 'IF' statement, for example:

```
if x>a
f=b
else
f=c
end
```

a smooth alternative is:

$$f(x) = \frac{1}{1+\mathrm{e}^{-2k(x-a)}}b + \frac{1}{1+\mathrm{e}^{2k(x-a)}}c, \tag{1}$$

where k is a constant that determines the 'sharpness' of the resulting function. This formulation works because:

$$\frac{1}{1+\mathrm{e}^{-2k(x-a)}} \sim \begin{cases} 1, \text{for } x > a \\ 0, \text{for } x < a \end{cases} \tag{2}$$

and

$$\frac{1}{1+\mathrm{e}^{2k(x-a)}} \sim \begin{cases} 0, \text{for } x > a \\ 1, \text{for } x < a \end{cases} \tag{3}$$

1.2 Smoothing 'MAX' and 'MIN' statements

MAX and MIN functions can be smoothed as follows[1]: Given a 'MAX' statement, for example:

[1] Development of smoothing practice in ADJULES was inspired by http:www.johndcook.comblog20100120how-to-compute-the-soft-maximum

`g=max(x,y)`
a smooth alternative (or soft maximum) is:

$$g(x,y;k) = \frac{\ln\left(\exp(k(x-j)) + \exp(k(y-j))\right)}{k} + j, \tag{4}$$

where k is a constant that determines the 'sharpness' of the resulting function and $j = \max(x,y)$. This formulation works because:

$$g(x,y;k) \sim \begin{cases} x, \text{for } x > y \\ y, \text{for } y > x \end{cases} \tag{5}$$

Similarly, given a '`MIN`' statement, for example:
`h=min(x,y)`
a smooth alternative (or soft minimum) is:

$$h(x,y;k) = \frac{-\ln\left(\exp(-k(x-j)) + \exp(-k(y-j))\right)}{k} + j, \tag{6}$$

where k is a constant that determines the 'sharpness' of the resulting function and $j = \max(x,y)$. This formulation works because:

$$h(x,y;k) \sim \begin{cases} y, \text{for } x > y \\ x, \text{for } y > x \end{cases} \tag{7}$$

The use of j to shift the arguments x and y avoids computational overflow or underflow.

2 Practice

In practice, only certain occurrences of these discontinuous functions need to be replaced with their smooth counterparts. In ADJULES, these are all `MAX` or `MIN` statements. For convenience, globally available functions called `softmin` and `softmax` have been written and can be used where a `MAX` or `MIN` statement is suspected to be causing a problem.

2.1 Smooth functions in ADJULES

`softmin` and `softmax` can be found in SOURCE/ADJULES/fomod.f90 and are written as follows:

```
function softmax(x,y,k) result(g)
real, intent(in) ::  x,y,k !  input
real ::  g !  output
real ::  j !maximum
if (l_smooth) then
j=max(x,y)
g = (log(exp(k*(x-j))+exp(k*(y-j))))/k+j
else
g = MAX(x,y)
end if
```

```
end function softmax


function softmin(x,y,k) result(h)
real, intent(in) ::  x,y,k !  input
real ::  h !  output
real ::  j !minimum
if (l_smooth) then
j=min(x,y)
h = -(log(exp(-k*(x-j))+exp(-k*(y-j))))/k+j
else
h = MIN(x,y)
end if
end function softmin
```

2.2 Use of the smooth functions in subroutines

Within JULES subroutines, smooth functions are applied as follows (from SOURCE/SUBROUTINES/SOIL/gauss.ff90):

```
x(i,1)=softmax(x(i,1),xmin(i,1),100.0)

x(i,1)=softmin(x(i,1),xmax(i,1),1000.0)
```

Note the use of the parameter k for sharpness. This can be adjusted to make a balance between accuracy (how close results from the smoothed code match results from the unsmoothed code) and smoothness (whether the code is practically differentiable or not).