Subject: IDL LSODE double precision
Posted by douglas Drumheller on Wed, 09 Aug 2017 19:37:51 GMT
View Forum Message <> Reply to Message

I've used LSODE to solve a simple oscillator problems successfully before in double precision. But for the first time I'm running on a 64-bit machine with Windows 7. One of my old codes starting acting up. So I've boiled it down to a test problem of a simple 2nd order linear harmonic oscillator.

```
PRO test_LSODE

y=[1.,0]

result=fltarr(100,2)

result(0,*)=y

time=findgen(100)/10.

for i = 1,99 do result(i,*) = lsode(y,0,time(i),'diff')

plot,time,result(*,0)

end

Function diff, t, y
  return, [y(1),-!pi^2*y(0)]
end
```

The plot will be a cosine of 5 complete cycles with a period of 2.

Problems start when you try to use double precision. In fact just changing the first statement to

```
y=[1.0D0,0.0D0]
```

will return garbage. I've tried setting the other input variables to double also, but the key is that the input y to lsode can no longer be double precision.

One of things I have noticed is that in single precision LSODE does not alter the input value of y. In double precision it does. I'd be happy with an explanation of LSODE doesn't support double precision, except it did on my old 32-bit machine.

Does anyone know what is going on?

Subject: Re: IDL LSODE double precision
Posted by chris_torrence@NOSPAM on Thu, 10 Aug 2017 20:10:38 GMT
View Forum Message <> Reply to Message

On Wednesday, August 9, 2017 at 1:37:56 PM UTC-6, douglas Drumheller wrote:
> I've used LSODE to solve a simple oscillator problems successfully before in double precision. But for the first time I'm running on a 64-bit machine with Windows 7. One of my old codes starting acting up. So I've boiled it down to a test problem of a simple 2nd order linear harmonic oscillator.
>
> PRO test_LSODE
>
> y=[1.,0]
>
> result=fltarr(100,2)
>
> result(0,*)=y
>
> time=findgen(100)/10.
>
> for i = 1,99 do result(i,*) = lsode(y,0,time(i),'diff')
>
> plot,time,result(*,0)
>
> end
>
> Function diff, t, y
>   return, [y(1),-!pi^2*y(0)]
> end
>
> The plot will be a cosine of 5 complete cycles with a period of 2.
>
> Problems start when you try to use double precision. In fact just changing the first statement to
>
> y=[1.0D0,0.0D0]
>
> will return garbage. I've tried setting the other input variables to double also, but the key is that the input y to lsode can no longer be double precision.
>
> One of things I have noticed is that in single precision LSODE does not alter the input value of y. In double precision it does. I'd be happy with an explanation of LSODE doesn't support double precision, except it did on my old 32-bit machine.
>
> Does anyone know what is going on?

Hi Douglas,

So I have a partial explanation. First, LSODE always converts the input argument to double precision, and then overwrites that variable with the result. If you pass in a double precision for Y, it doesn't need to convert it and it just changes the values.

I can't explain why the behavior would have changed between your 32 and 64-bit machines. I can reproduce the problem here, but I don't know why it's happening. One workaround is to reduce the step size by using the previous value and a constant step size. For your example:

```
  for i = 1,99 do begin
    y1 = result[i-1,*]
    result[i,*] = lsode(y1,i-1,0.1d,'diff',status)
  endfor
```

This seems to work fine for double-precision input.

Hope this helps,
Chris

Subject: Re: IDL LSODE double precision
Posted by douglas Drumheller on Sat, 12 Aug 2017 16:57:53 GMT
View Forum Message <> Reply to Message

Thanks Chris,

It's helpful to understand that LSODE's normal mode is double precision. Your code rewrites the input y before each use, which seems to do the trick. I'm studying the effect of weak nonlinearities on the period of the oscillation, and I iterate on parameters which alter this nonlinearity, which is why I integrate from the initial value of time rather than incrementally as you suggest.

Doug
