Subject: Re: Can this be vectorized?
Posted by davis on Tue, 26 Oct 1999 07:00:00 GMT
View Forum Message <> Reply to Message

On 26 Oct 1999 09:08:26 GMT, Struan Gray <struan.gray@sljus.lu.se>
wrote:
> In IDL an alternative to vectorisation is to use the fast built-in
> function HISTOGRAM. In your case you should take the histogram of 'I'
> and use the REVERSE_INDICES keyword to get a list of which elements
> are in which bin. Summing the same elements of 'X' will give you the
> answer you want. This will work even if 'I' and 'X' are not in
> ascending order. There is some memory overhead, but it is of the same
> order as creating working copies of your original data.

Let's suppose that the integer array `I' is:

```
I = [0, 1, 1, 2, 3, 4, 4, 4, 5]
```

and `X' = [a, b, c, d, e, f, g, h, i].

Then, I want `Y' to be:

```
Y = [a, (b+c), d, e, (f+g+h), i]
```

The histogram `H' is:

```
H = [1, 2, 1, 1, 3, 1]
```

and the REVERSE_INDICES array `R' is:

```
R = [7, 8, 10, 11, 12, 15, 16, 0, 1, 2, 3, 4, 5, 6, 7, 8]
```

It seems to me that I would have to loop over all terms in the
histogram via (pseudocode!):

```
H = H[where(H != 0)]          # get rid of non-zero elements
j = 0
for i = 0 to i = length(H)
  if R[i] != R[i+1]
    J = [ R[i]:R[i+1]-1 ]
    Y[j] = sum (X[J])
  endif
next i
```

Of course this is useful as long as `I' contains many repeated
elements so that the histogram will be small. Unfortunately, in my
case this is unlikely.

At least I have a better understanding of the REVERSE_INDICES keyword.
Thanks,
--John

Subject: Re: Can this be vectorized?
Posted by Struan Gray on Tue, 26 Oct 1999 07:00:00 GMT
View Forum Message <> Reply to Message

John E. Davis, davis@space.mit.edu writes:

> What is the best way to vectorize this?

In IDL an alternative to vectorisation is to use the fast built-in function HISTOGRAM. In your case you should take the histogram of 'I' and use the REVERSE_INDICES keyword to get a list of which elements are in which bin. Summing the same elements of 'X' will give you the answer you want. This will work even if 'I' and 'X' are not in ascending order. There is some memory overhead, but it is of the same order as creating working copies of your original data.

Struan

Subject: Re: Can this be vectorized?
Posted by Gautam Sethi on Tue, 26 Oct 1999 07:00:00 GMT
View Forum Message <> Reply to Message

In comp.soft-sys.matlab Bert Jagers <hrajagers@my-deja.com> wrote:
: Dear John,

: Two solutions in Matlab: one completely vectorized, one partially.
: Of course, you could also implement it as a MEX file.

: F=find([1 diff(I) 1]);
: XS=[0 cumsum(X)];
: Y=XS(F(2:end))-XS(F(1:(end-1)));

: There is one major drawback to this implementation, since

:> In reality, X consists of about one million elements,

: you may loose accuracy when taking the difference of two large
: cummulative values. So, I tried to find another solution.

: F=find([1 diff(I) 1]);

: Y=zeros(1,length(F)-1);
: for i=1:length(Y),
:   Y(i)=sum(X(F(i):(F(i+1)-1)));
: end;

: The memory usage is probably comparable. In both cases there needs to
: be space for the matrices I,X,F and [1 diff(I) 1], or else Matlab will
: start swapping.

: Best regards,

: Bert Jagers

since memory is of issue, you may want to try this non-vectorized one as well:
 ---------- ------
function K = davis(I,J)

for i = min(I):max(I)
   K(i) = sum(J(find(I == i)));
end
 ---------- ------

Subject: Re: Can this be vectorized?
Posted by Bert Jagers on Tue, 26 Oct 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Dear John,

Two solutions in Matlab: one completely vectorized, one partially.
Of course, you could also implement it as a MEX file.

F=find([1 diff(I) 1]);
XS=[0 cumsum(X)];
Y=XS(F(2:end))-XS(F(1:(end-1)));

There is one major drawback to this implementation, since

> In reality, X consists of about one million elements,

you may loose accuracy when taking the difference of two large
cummulative values. So, I tried to find another solution.

F=find([1 diff(I) 1]);
Y=zeros(1,length(F)-1);
for i=1:length(Y),
  Y(i)=sum(X(F(i):(F(i+1)-1)));
end;

The memory usage is probably comparable. In both cases there needs to
be space for the matrices I,X,F and [1 diff(I) 1], or else Matlab will
start swapping.

Best regards,

Bert Jagers

Sent via Deja.com http://www.deja.com/
Before you buy.

Subject: Re: Can this be vectorized?
Posted by davis on Wed, 27 Oct 1999 07:00:00 GMT
View Forum Message <> Reply to Message

On 26 Oct 1999 07:00:37 GMT, Gautam Sethi <gautam@nospam.are.berkeley.edu>
wrote:
> function K = davis(I,J)
>
> for i = min(I):max(I)
>   K(i) = sum(J(find(I == i)));
> end

I tried something like this earlier but the result was slower than
the unvectorized version.

Thanks,
--John
