
Subject: Re: discrete cosine transform

Posted by [Peter Mason](#) on Fri, 17 Dec 1999 08:00:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

kremasti@sbox.tu-graz.ac.at (eva) wrote:

> Im looking for an algorithm for the discrete cosine transform (and
> for the inverse discrete cosine transform) ,
> can anybody help ?

Hi Eva,

Well it's been a couple of days now and no tug on the line; let's see if my little arm-waving interlude here will coax out a response from someone who knows their stuff :-)

The discrete cosine transform (DCT) of a 1D real function "FUNC" is really little more than a discrete fourier transform (DFT) of FUNC after it has been rigged to be even (symmetrical about the Y axis). Because the rigged function is even, the sine terms (the imaginary part) of its fourier transform are all zero, leaving only the (real) cosine terms. Although there are routines to do a DCT directly, you can therefore also get it via a DFT by doubling up the function to make it even, doing DFT, tossing out the imaginary part, and doing a bit of scaling. Now there's more than one way to double up a 1D real function, but I gather that one particular way is accepted in practice. I think I have it implemented in my example code below, but I'm not entirely certain. Anyway, appended is an impersonation of a 1D forward and reverse DCT routine. There's a scaling factor of 2 missing somewhere that I haven't been able to track down (i.e., the zeroth term is supposed to be the function's average but it's off by a factor of 2), but the transform *is* properly reversible.

If you want a 2D DCT of a matrix MAT, you can apply this routine across the rows of MAT to get MAT_1, and then apply it down the columns of MAT_1 to get MAT_DCT (as you could do with a 1D fourier transform routine to get a 2D fourier trnasform). If you did this, though, you'd probably want to move some of the calculations out of the routine below in order to do them only once.

Cheers

Peter Mason

ICQ: 29778826

```
;=====
function sad_FCT,arr,direc
;Slow-and-dirty 1D Cosine Transform (but real and reversible)
;Set direc=-1 for a forward transform and direc=1 for reverse.
n=n_elements(arr)
b=reform(arr,n)
even_el=lindgen(long(n-1)/2+1)*2
```

```

odd_el=rotate((even_el+1),2)
if ((n mod 2) ne 0) then odd_el=odd_el(1:n_elements(odd_el)-1)
if (direc lt 0) then begin ;FORWARD
  w=2.0*exp((cindgen(n)*complex(0,-1)*!pi)/(2.0*n))
  b=complex(b([even_el,odd_el]))
  b=fft(b,-1,/overwrite)
  return,float(b*w)
endif else begin ;REVERSE
  w=0.5*exp((cindgen(n)*complex(0,1)*!pi)/(2.0*n))
  c=[0.0,b(n-1-lindgen(n-1))]
  b=complex(b,-c)*w
  b=fft(b,1,/overwrite)
  j=fltarr(n)
  j([even_el,odd_el])=float(b)
  return,j
endelse
end
;=====

```

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