
Subject: Speeding up xy distance calculations using complex numbers

Posted by [kagoldberg](#) on Sat, 18 May 2013 09:16:53 GMT

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

This may be well known to many people, but I just discovered that in IDL, you can greatly speed up x,y distance calculations using complex numbers to represent positions in the xy plane, in some cases.

The example below takes you through four nominally identical calculations in which we define a square array of x and y values (or complex $z=x+iy$ values), and for each point, compute the square of the $(dx^2 + dy^2)$ distance from a given point, $p=(x,y)$ or $zp=x+iy$. For the complex calculations I use $d * \text{conj}(d)$. I think that's the fastest available way of doing this.

I found that for array sizes below approximately 300x300, the complex method can be 2x faster. They even out when the array sizes become larger, or the complex calculation takes slightly longer. I have examples here for both single and double-precision. Results may depend on your hardware, I presume.

for N=100,500,100 do begin ;--- N is the width of the square arrays

```
print, '---- N = ', N
```

```
    ;--- perform tests with single-precision
```

```
    x = findgen(N) # (1. + fltarr(N))
```

```
    y = (1. + fltarr(N)) # findgen(N)
```

```
    z = x + complex(0,1)*y
```

```
    p = [0.5,0.5]
```

```
    t0 = systime(1)
```

```
    for i=0,99 do $
```

```
        distance_squared = (x-p[0])^2 + (y-p[1])^2
```

```
    print, 'FLOAT Real distance calculation: ', (systime(1) - t0)/100.
```

```
    zp = p[0] + complex(0,1)*p[1]
```

```
    t0 = systime(1)
```

```
    for i=0,99 do begin
```

```
        z1 = z - zp
```

```
        distance_squared = z1 * conj(z1)
```

```
    endfor
```

```
    print, 'FLOAT Complex distance calculation: ', (systime(1) - t0)/100.
```

```
    ;--- perform this with double-precision
```

```
    x = dindgen(N) # (1. + dblarr(N))
```

```
    y = (1d + dblarr(N)) # dindgen(N)
```

```
    z = x + dcomplex(0,1)*y
```

```
    p = [0.5d,0.5d]
```

```
t0 = systime(1)
for i=0,99 do $
  distance_squared = (x-p[0])^2 + (y-p[1])^2
print, 'DOUBLE Real distance calculation: ', (systime(1) - t0)/100.

zp = p[0] + dcomplex(0,1)*p[1]
t0 = systime(1)
for i=0,99 do begin
  z1 = z - zp
  distance_squared = z1 * conj(z1)
endfor
print, 'DOUBLE Complex distance calculation: ', (systime(1) - t0)/100.
endfor

end
```
