Complex Sum

In this example, we use the functions on screen 9 to do complex arithmetic and the sum key on screen 7 to evaluate a complex summation.

Find the approximation to $f(z) = e^z$ that comes from adding a finite number of its Taylor series terms:

$$e^z \approx 1 + z + \frac{z^2}{2!} + \frac{z^2}{3!} + \cdots + \frac{z^n}{n!}$$

We want to use the sum key for the summation, but as a screen 7 function, sum adds only real numbers. So we will accumulate the sum as a complex number and keep it in registers 11,12 (the complex store and recall keys use two registers for the real and imaginary parts of a complex number).

The sum function will loop over the terms in the sum, but we will ignore the sum key's output and recall our complex sum at the end.

Define function f1(k) to compute the current term $z^k/k!$, recall the partial sum, add the current term, then put the updated partial sum back into registers 11,12.

```
f1: 9, func, 2, sto, 0, rcl, x↔y, y^x,
    3, func, 2, rcl, x!,
    9, func, /, 11, rcl, +, 11, sto
```

The input to f1 is k, and f1 saves it in registers 2,3 (register 3 will be zero, since k is a real integer).
z will have been stored in registers 0,1 before starting the sum.

Function f2 has 2 inputs, n (number of terms to sum) and z (value at which to evaluate the sum).

```
f2: 9, func, 0, sto, roll, 4, sto,             (store z in registers 0,1, and n in register 4)
    0, enter, 11, sto,                         (initialize the sum to zero in registers 11,12)
    7, func, 0, enter, 4, rcl, 1, enter, 1, sum (sum of f1 for k from 0 to n by steps of 1)
    9, func, 11, rcl                           (recall the sum from registers 11,12)
```

When we switch between the complex screen and a real screen to get x! in f1 and sum in f2, the calculator tries to do the right thing so the calculation works as we intended.

For f1, when k is stored into registers 2,3 by the "2, sto" command in screen 9, k will actually be a real integer, so k is in register 2 and zero is in register 3. The complex store command shows as "zsto" in the f1 program listing on the screen. Then the "2, rcl" command from screen 3 is in real mode and only register 2 is used. The x! is also in real mode. The division by x! is back in complex mode, and since the x-register on the stack was defined in real mode, it has a zero imaginary part for the complex division.

The result of all this is that we don't have to do anything special when using real-valued operations as part of a complex-valued function.

Approximate $e^{1+2\,i}$ by summing the series to the $z^{30}/30!$ term.
On screen 9 enter the complex number $1+2i$ by pressing: 1, +i, 2.

9, func, 12, fix, 30, enter, $1+2i$, enter, 2, $\mathrm{f_n}$

This gives $-1.131204383757 + 2.471726672005$ i.

Check by subtracting $e^{1+2\,i}$.

$1+2i$, $\mathrm{e^x}$, $-$

This gives 8.424054458174829742e−24 - 1.459883242583510960e−24 i.
So using 30 terms of the Taylor series gave about 24 digits of accuracy for this z.

Try the series again for $z = e^{1+2\,i}$, this time using 50 terms:

50, enter, $1+2i$, enter, 2, $\mathrm{f_n}$

This again shows $-1.131204383757 + 2.471726672005$ i

Check by subtracting $e^{1+2\,i}$.

$1+2i$, $\mathrm{e^x}$, $-$

This gives −4.372820700000000000e−49 + 1.965364000000000000e−50 i, so 50 terms of the Taylor series gives about 49 digits of accuracy for this z.

Fewer digits can be displayed for complex numbers, but this shows that even though the 30-term and 50-term results from f2 look the same in the display, they still have over 50-digit precision internally.