Julia Inverse **I**terative **M**ethod

Alain Brobecker

1. Squareroot of a complex number

Given $a+ib \in \mathbb{C}$ we search for $x+iy \in \mathbb{C}$ such that $a+ib=(x+iy)^2$.

$$a+ib=(x+iy)^2 \iff a+ib=x^2+2ixy-y^2 \iff \begin{cases} x^2-y^2=a \\ 2xy=b \end{cases}$$

$$\iff \begin{cases} x^2=a+y^2 \\ 4x^2y^2=b^2 \\ sgn(x)\times sgn(y)=sgn(b) \end{cases} \iff \begin{cases} x^2=a+y^2 & \\ 4y^4+4ay^2-b^2=0 & (1) \\ sgn(x)\times sgn(y)=sgn(b) & (2) \end{cases}$$

To solve (1) we compute the discriminant $\Delta=16a^2+16b^2>0$ and then

$$y^2=\frac{-4a\pm\sqrt{16a^2+16b^2}}{8}=\frac{-a\pm\sqrt{a^2+b^2}}{2}$$

Since $\sqrt{a^2+b^2}>|a|$ and since $y^2>0$ we have:

$$y^2=\frac{\sqrt{a^2+b^2}-a}{2} \quad , \quad x^2=\frac{\sqrt{a^2+b^2}+a}{2}$$

Mixing this solution of (1) with (2) we finally conclude that:

$$\boxed{a+ib=(x+iy)^2 \iff \begin{cases} x=\pm\sqrt{\dfrac{\sqrt{a^2+b^2}+a}{2}} \\ y=\pm\sqrt{\dfrac{\sqrt{a^2+b^2}-a}{2}} \\ sgn(x)\times sgn(y)=sgn(b) \end{cases}}$$

Notes:

► If $x \neq 0$ (which is a sure thing if $b \neq 0$) we can use $y=\dfrac{b}{2x}$ in order to replace a squareroot and sign computation by a division.

► If $(x+iy)^2=a+ib$ then $(-y+ix)^2=-a-ib$.

▷ If $a=b=0$ then $x=y=0$.

▷ If $a=0$ and $b\neq 0$ then $x=\pm\sqrt{\dfrac{|b|}{2}}$ and $y=\pm\sqrt{\dfrac{|b|}{2}}$ in respect to (2).

▷ If $b=0$ and $a>0$ then $x=\pm\sqrt{a}$ and $y=0$.

▷ If $b=0$ and $a<0$ then $y=\pm\sqrt{-a}$ and $x=0$.

2. Application to Julia IIM

For a given polynomial complex function f the Julia set of f is the boundary of the set of points which converge to infinity under iteration. The functions generally drawn on computer are $f(z) = z^2 + c$.
Generally, for every point $M \in \mathbb{C}$ of the screen we compute $f^k(M)$ for $k < n$. As soon as $|f^k(M)| > 1$ we know it will diverge and we print it on screen with color k.
But the Julia set can also be obtained by backward iteration, ie the Julia set is the set of limit points of $\bigcup_{n\in\mathbb{N}} f^{-n}(z)$.
So we need to compute iterations of $f^{-1}(z) = \sqrt{z - c}$, and since this has two solutions the number of branches at iteration n will be 2^n. But for practical implementation we stop computing a branch if the point approximated by $f^{-k}(z)$ is already drawn on screen.

3. Algorithm

```
  oldx=0
  oldy=0
  NbPointsInStack=0
  GOTO ProcessOnePoint

GetPointFromStack
  if NbPointsInStack=0 then END
  get (oldx;oldy) from stack
  NbPointsInStack-=1
ProcessOnePoint
  x=oldx-cx
  y=oldy-cy
  t=sqrt((x+sqrt(x*x+y*y))/2)
  if t=0 then
    newy=sqrt(-x)
    newx=0
  else
    newx=t
    newy=y/(2*newx)
  endif
  if (newx;newy) is already drawn then GOTO GetPointFromStack
  if NbPointsInStack>Threshold then draw (newx;newy) and (-newx;-newy)
  put (newx;newy) in stack
  NbPointsInStack+=1
  oldx=-newx
  oldy=-newy
  GOTO ProcessOnePoint
```