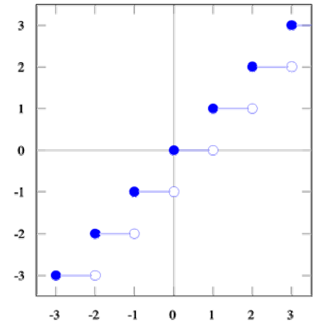
Two Effects in Real Computation

a) Multivalued 'functions'



Example (Archimedian Reals):
Given $x \in \mathbb{R}$,
return <u>some</u> integer upper bound.

Example (Fundamental Theorem of Algebra):
Given $a_0, \ldots a_{d-1} \in \mathbb{C}$, return roots $x_1, \ldots x_d \in \mathbb{C}$ of $a_0 + a_1 \cdot X + \ldots + a_{d-1} \cdot X^{d-1} + X^d \in \mathbb{C}[X]$ incl. multiplicities

up to permutation [Specker'67]

b) Discrete 'advice'

Example matrix diagonalization: given $A \in \mathbb{R}^{d \cdot (d-1)/2}$,
return a basis of eigenvectors — discontinuous:

Thm: Computable knowing $|\sigma(A)|$. $\varepsilon \cdot \begin{pmatrix} \cos(2/\varepsilon) & \sin(2/\varepsilon) \\ \sin(2/\varepsilon) & -\cos(2/\varepsilon) \end{pmatrix}$

Exact Real Computation

- *Imperative* programming paradigm [Müller/Z.14]
- Implemented in C++ library **`iRRAM`** [Müller'01] with data type **`REAL`** and operator overloading
- Operate on *arguments exactly*, return *approximation* to *value* up to absolute error 2^{p}.

In numerics, don't test for in/equality!

- *Partial* semantics of tests: „**`x>y`**" = ↑ if $x=y$.
- *Non*-extensional so-called *Parallel-OR* [EHS'04]
  choose(x_1>y_1,…,x_d>y_d) returns *some* index j with $x_j > y_j$, provided such exists.

Examples in ERC

Example 0: *Fuzzy/soft* test [Sagraloff,C.Yap,...]

choose ($x+2^{-n}>y$, $y+2^{-n}>x$)

Example 1: Round: $\mathbb{R} \ni r \to z \in \mathbb{Z}$ s.t. $r-1 < z \le r+1$

```
INTEGER Round (REAL r);
INTEGER z:=0; REAL x:=r;
WHILE  choose( z < r , z > r-1 ) == 1  DO
   z := z + 1;
RETURN z
```

- *Partial* semantics of tests: „**x>y**" = ↑ if $x=y$.
- *Non*-extensional so-called *Parallel-OR* [EHS'04]

choose($x_1>y_1$,…,$x_d>y_d$) returns *some* index j with $x_j>y_j$, provided such exists.

Faster Non-extensional Integer Rounding

Example 1: Round: $\mathbb{R} \ni r \to z \in \mathbb{Z}$ s.t. $r-1 < z \le r+1$

```
INTEGER Round (REAL r);
INTEGER z:=0; REAL x:=r;  INTEGER b; INTEGER k:=0;
WHILE choose( |x|> ½ , |x| < 1 ) == 1 DO
  k:=k+1;   x:=x/2;  END;  // normalize x
WHILE k>0 DO ; x := x·2;
  b := choose( x<0 , -1<x<1 , x>0 ) - 2;
  // most signif. signed bin. digit -1,0,+1 of x
  x := x - b;  z := z + z + b;  k := k - 1;
  END; RETURN z;
```

- *Partial* semantics of tests: „**x>y**" = ↑ if $x=y$.
- **choose**($x_1>y_1$,…,$x_d>y_d$) = *some* j s.t. $x_j>y_j$

Trisection for Simple Root Finding


```
REAL Trisection (INTEGER p; C([0;1],REAL) f);
// f(x) < 0 < f(y),    ∃! z∈[0,1]: f(z)=0
REAL x:=0; REAL y:=1;
WHILE choose( y−x > 2^(p−1) , 2^p > y−x ) == 1  DO
  IF choose ( 0>f((2x+y)/3) , 0<f((x+2y)/3) ) == 1
   THEN  x:=(2x+y)/3;  ELSE  y:=(x+2y)/3;  ENDIF;
END; RETURN x;
```

- *Partial* semantics of tests: „**x>y**" = $\uparrow$ if $x=y$.
- **choose**($\mathbf{x}_1>\mathbf{y}_1,\ldots,\mathbf{x}_d>\mathbf{y}_d$) = *some* j s.t. $x_j>y_j$