NewtonGF[NumericalNewtonIteration] - returns a procedure to compute numerical values of generating functions by Newton iteration (oracle in the Boltzmann sampling method)

Calling Sequence

NumericalNewtonIteration(**Sys**, **labelling**)

Parameters

`Sys` – set of equations; a grammar in the combstruct syntax
`labelling` – one of labelled, labeled, unlabelled, unlabeled, as in combstruct

Description

- The **NumericalNewtonIteration** command returns a procedure that takes as input a nonnegative real number and returns the values of the generating series at that point.
- The procedure first computes the decomposition of the system into strongly connected components and then generates a Newton iteration for each of them.
- This command is part of the **NewtonGF** package, so it can be used in the form **NumericalNewtonIteration(..)** only after executing the command **with(NewtonGF)**. However, it can always be accessed through the long form of the command by using **NewtonGF**[**NumericalNewtonIteration**](..).

Examples

```
> with(NewtonGF);
```

$[BoltzmannExpectedSize, BoltzmannParameter, GFSeries, NumericalNewtonIteration, Radius, SeriesNewtonIteration]$ **(2.1)**

A grammar for series-parallel circuits.

```
> circuit:={C=Union(P,S,R), P=Set(Union(S,R),card>=2), S=Set
  (Union(P,R),card>=2), R=Atom};
```

$circuit := \{C = Union(P, S, R), P = Set(Union(S, R), 2 \leq card), R = Atom, S = Set(Union(P, R), 2 \leq card)\}$ **(2.2)**

Here are the corresponding equations over generating functions:
combstruct[gfeqns](circuit, labeled, z);

$$[C(z) = P(z) + S(z) + z, P(z) = e^{S(z)+z} - 1 - S(z) - z, R(z) = z, S(z) = e^{P(z)+z} - 1 - P(z) - z]$$

```
> oracle:=NumericalNewtonIteration(circuit, labeled);
```

$oracle := \textbf{proc}(x, \{ending\_block::integer := 3\})$ ... **end proc** **(2.3)**

```
> oracle(0.1);
```

$[C = 0.1115989517, P = 0.005799475874, R = 0.1, S = 0.005799475874]$ **(2.4)**

```
> oracle(0.3);
```

$[C = 0.4685471294, P = 0.08427356468, R = 0.3, S = 0.08427356468]$ **(2.5)**

Newton's iteration does not converge outside the radius of convergence of the generating function (0.3862943611 here).

```
> oracle(0.4);
Error, (in recursivenewton) One coordinate at least has
```

```
decreased, Array(1..2, {(1) = .3024050494542, (2) =
.3024050494542}), Array(1..2, {(1) = -.4390112696851, (2) =
-.4390112696851})
```

The unlabeled case with Sets or Cycles are also implemented:

```
> oracle:=NumericalNewtonIteration(circuit, unlabeled);
```

$$oracle := \textbf{proc}(x, \{ending\_block::integer := 3\}) \ \ldots \ \textbf{end proc} \tag{2.6}$$

```
> oracle(0.1);
```

$$[C = 0.1253314199, P = 0.01266570996, R = 0.1, S = 0.01266570996] \tag{2.7}$$

In the unlabelled case, the radius is about 0.2808326670.

```
> oracle(0.3);
Error, (in recursivenewton) Newton iteration does not seem to
converge
> oracle(0.2808);
```

$$[C = 0.9842607881, P = 0.3517303941, R = 0.2808, S = 0.3517303941] \tag{2.8}$$

See Also

combstruct[gfseries], NewtonGF[SeriesNewtonIteration], NewtonGF[Radius], NewtonGF