Fachhochschule Vorarlberg

Algorithmen für Graphen

Effective Branching Factor

Patrick Ritschel, Version 1.1

Mathematik
Newtonsches Näherungsverfahren

- Dient der Nullstellensuche (findet *eine* NST)
- Arbeitet iterativ
  - das Ergebnis wird schrittweise verbessert (angenähert)
  - gestartet wird mit einem Wert, möglichst nahe der vermuteten Nullstelle
- Legt eine Tangente durch *f(x)*, schneidet diese mit der x-Achse und bestimmt so das neue *x*

$$x_{n+1} := N_f(x_n) := x_n - \frac{f(x_n)}{f'(x_n)}$$

Mathematik
Newtonsches Näherungsverfahren

- *z.B.: $f(x) = cos(x) – x^3$*

```
function f(x) {
  return cos(x) - x³
}
function f‛(x) {
  return -sin(x) - 3x²
}
function NewtonIterationFnct(x) {
  return x - f(x) / f‛(x)
}
x := 0.5 // Geratener Startwert
do {
  xold := x
  x := NewtonIterationFnct(x)

} while (abs(xold - x) > SCHRANKE) // z.B. SCHRANKE == 0.001
```

Mathematik

Effective Branching Factor

Fachhochschule Vorarlberg

- Anzahl der Knoten im Baum:

  $b^0+b^1+\ldots+b^d = n$

- In geschlossene Form reduzieren:

  $$b^1+\ldots+b^d+b^{(d+1)} = bn$$
  $$b^0+b^1+\ldots+b^d \qquad = n$$

  $$-(b^0) \qquad +b^{(d+1)} = bn - n$$

- Also erhalten wir (und definieren)

  $b^{(d+1)} - 1 - bn + n = 0 =: f(b)$

- Und deshalb [Lsg Newtonsche Näherung, b_0 = 2.0]

  $f'(b) = (d+1)b^d - n$