Mehrgitterverfahren

```
> restart;
> with(plots):
Warning, the name changecoords has been redefined
```

Hilfsfunktionen: Bilder malen und Startwerte erzeugen

`bild` malt eine Liste von Punkten ueber dem Einheitsintervall

```
> bild := proc(u)
  local i,n;
  n := nops(u);
  plot([seq([i/(n+1),u[i]], i=1..n)],style=line)
  end;
```

$$bild := \mathbf{proc}(u) \quad \mathbf{(1.1)}$$

$$\mathbf{local}\ i, n;$$

$$n := nops(u);\ plot([seq([i/(n+1), u[i]], i=1..n)], style=line)$$

$$\mathbf{end\ proc}$$

`bilder` malt eine Liste von Bildern wie oben in ein Diagramm

```
> bilder := proc(l)
  local u,pl;
  pl := {};
  for u in l do pl := pl union {bild(u)} od;
  display(pl)
  end;
```

$$bilder := \mathbf{proc}(l) \quad \mathbf{(1.2)}$$

$$\mathbf{local}\ u, pl;$$

$$pl := \{\};\ \mathbf{for}\ u\ \mathbf{in}\ l\ \mathbf{do}\ pl := union(pl, \{bild(u)\})\ \mathbf{end\ do};\ display(pl)$$

$$\mathbf{end\ proc}$$

```
> rn1 := rand(2000);
  rn := 'evalf(rn1()-1000)/1000';
```

$$rn1 := \mathbf{proc}(\,) \quad \mathbf{(1.3)}$$

$$\mathbf{local}\ t;$$

$$\mathbf{global}\ \_seed;$$

$$\_seed := irem(a * \_seed, p);$$

$$t := \_seed;$$

$$\mathbf{to}\ concats\ \mathbf{do}\ \_seed := irem(a * \_seed, p);\ t := s * t + \_seed\ \mathbf{end\ do};$$

$$irem(t, divisor) + offset$$

$$\mathbf{end\ proc}$$

$$rn := \frac{1}{1000}\ evalf(rn1(\,) - 1000)$$

Residuum zum Differenzenstern $\frac{[-1, 2, -1]}{h^2}$

```
> residuum := proc(x,b)
  # option trace;
  local i,n,hh,tmp;
  n := nops(x);
  hh := 1/(n+1)^2;
  tmp := [0, op(x), 0]; # Randbedingungen
  [seq(b[i] - (2*tmp[i+1] - tmp[i] - tmp[i+2])/hh, i=1..n)]
  end;
```

$$residuum := \mathbf{proc}(x, b)$$

$$\mathbf{local}\ i, n, hh, tmp;$$

$$n := nops(x);$$

$$hh := 1 / (n + 1)\,\hat{}\,2;$$

$$tmp := [0, op(x), 0];$$

$$[seq(b[i] - (2 * tmp[i + 1] - tmp[i] - tmp[i + 2]) / hh, i = 1 ..n)]$$

$$\mathbf{end\ proc}$$

(2.1)

Jacobi-Iteration (gedaempft mit 1/2)

```
> jacobi := proc(x,b)
  # option trace;
  local r,n,hh,i;
  n := nops(x);
  hh := 1/(n+1)^2;
  r := residuum(x,b);
  1/2*(hh/2)*r+x, b
  end;
```

$$jacobi := \mathbf{proc}(x, b)$$

$$\mathbf{local}\ r, n, hh, i;$$

$$n := nops(x);\ hh := 1 / (n + 1)\,\hat{}\,2;\ r := residuum(x, b);\ 1 / 4 * hh * r + x, b$$

$$\mathbf{end\ proc}$$

(3.1)

Restriktion der rechten Seite auf das $2h$-Gitter

Hier: triviale Restriktion

```
> restriktion := proc(b)
  local n,bg,i;
  n := (nops(b)+1)/2-1;
  bg := [seq(b[2*i], i=1..n)]
  end;
```

$$restriktion := \mathbf{proc}(b) \quad \textbf{(4.1)}$$

$$\mathbf{local}\ n, bg, i;\ n := 1/2 * nops(b) - 1/2;\ bg := [seq(b[2*i], i = 1..n)]$$

$$\mathbf{end\ proc}$$

Prolongation der Korrektur auf das $\frac{h}{2}$-Gitter

Hier: lineare Interploation

```
> prolongation := proc(x)
  local xf,n,i;
  n := nops(x);
  xf := [0, seq(op([x[i],0]), i=1..n)];
  for i from 2 by 2 to 2*n do
    xf[i-1] := xf[i-1] + xf[i]/2;
    xf[i+1] := xf[i+1] + xf[i]/2
  od;
  xf
  end;
```

$$prolongation := \mathbf{proc}(x) \quad \textbf{(5.1)}$$

$$\mathbf{local}\ xf, n, i;$$

$$n := nops(x);$$

$$xf := [0, seq(op([x[i], 0]), i = 1..n)];$$

$$\mathbf{for}\ i\ \mathbf{from}\ 2\ \mathbf{by}\ 2\ \mathbf{to}\ 2*n\ \mathbf{do}$$

$$xf[i-1] := xf[i-1] + 1/2 * xf[i];\ xf[i+1] := xf[i+1] + 1/2 * xf[i]$$

$$\mathbf{end\ do};$$

$$xf$$

$$\mathbf{end\ proc}$$

Das Mehrgitterverfahren

- x: Startvektor (oder [], falls Startwert [0...0])
- b: rechte Seite

- l: Anzahl der Vergroeberungsschritte
- nu1, nu2: Vor- bzw. Nachglaettungen

```
> mg := proc(x, b, l, nu1, nu2)
  local i, xneu, xg, tmp;
  # option trace;
  xneu := x;
  if xneu=[] then xneu := [seq(0, i=1..nops(b))] fi;
  xneu:= (jacobi@@nu1)(xneu,b)[1];
  if l>0 then
    xneu := xneu + prolongation(mg([], restriktion(residuum
  (xneu,b)), l-1, nu1, nu2)[1]);
  fi;
  (jacobi@@nu2)(xneu, b), l, nu1, nu2
  end;
```

$$mg := \mathbf{proc}(x, b, l, nu1, nu2) \quad \textbf{(6.1)}$$

$\mathbf{local}\ i, xneu, xg, tmp;$

$xneu := x;$

$\mathbf{if}\ xneu = [\,]\ \mathbf{then}\ xneu := [seq(0, i = 1..nops(b))]\ \mathbf{end\ if};$

$xneu := `@@`(jacobi, nu1)(xneu, b)[1];$

$\mathbf{if}\ 0 < l\ \mathbf{then}$

$xneu := xneu + prolongation(mg([\,], restriktion(residuum(xneu, b)), l - 1, nu1, nu2)[1])$

$\mathbf{end\ if};$

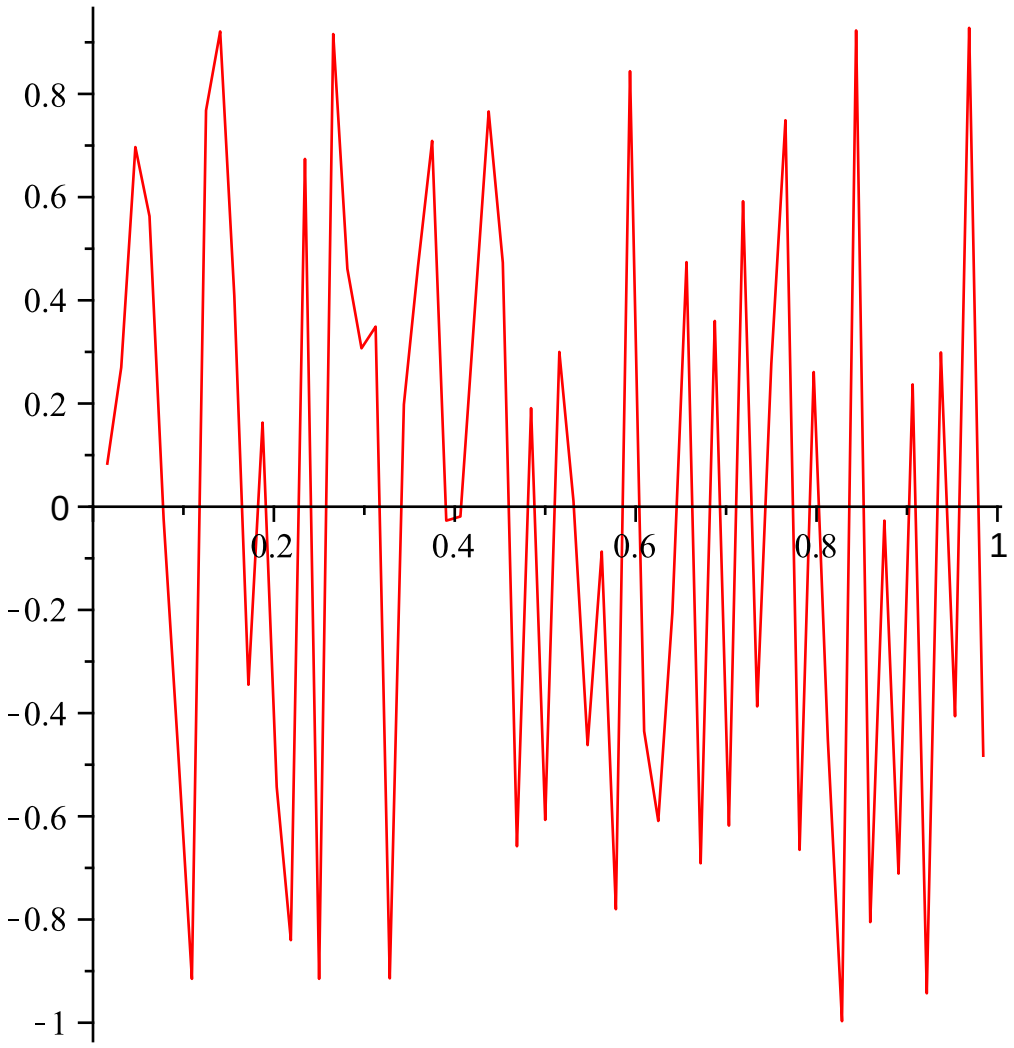
$`@@`(jacobi, nu2)(xneu, b), l, nu1, nu2$

$\mathbf{end\ proc}$

Wir loesen mit rechter Seite 0 und 'zufaelligen' Startwerten

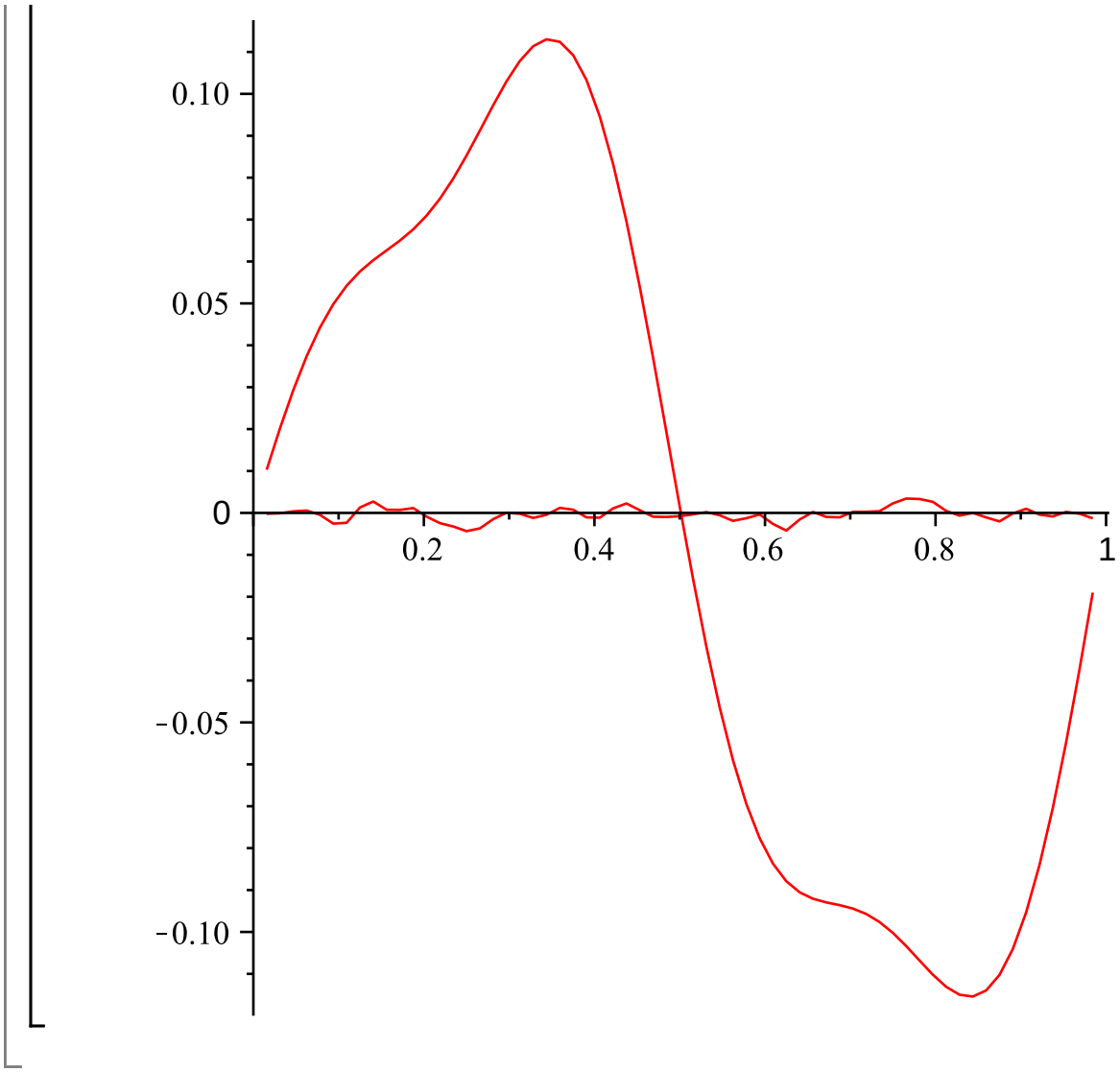
```
> l := 6:
> n := 2^l-1;
> x0 := [seq(rn(), i=1..n)]:
> b := [seq(0, i=1..n)]:
> bild(x0);
```

$$n := 63$$



Im Vergleich: Jacobi und Mehrgitter

```
> bilder({[(jacobi@@100)(x0,b)][1],[(mg@@2)(x0,b,l,2,2)][1]});
```


0.10
0.05
0
-0.05
-0.10
0.2
0.4
0.6
0.8
1