Protokoll zum Zusatzversuch KdV-Gleichung und SQM

Jan Steinhoff
Betreuer: Prof. Wipf

8. März 2005

Inhaltsverzeichnis

1 Aufgabenstellung

Mit Hilfe der supersymmetrischen Quantenmechanik (SQM) kann zum Potential $V(x) = -n(n+1)\,\mathrm{sech}^2\,x$ eine n-parametrige Schar $V(\lambda_1, ..., \lambda_n; x)$ von Potentialen gefunden werden, die das gleiche Spektrum wie $V(x)$ haben. Durch eine geeignete Parametertransformation zwischen den λ_i und der Zeit t erhält man eine n-Solitonenlösung der Korteweg-de Vries-Gleichung ([1], Abschnitt 7.3).

1. Finde die Parametertransformation für $n = 1$.
2. Finde die Parametertransformation für $n = 2$.
3. Numerisches Lösen der KdV-Gleichung mit zwei Solitonen und Vergleich mit der analytischen Lösung.

2 Grundlagen

2.1 Supersymmetrische Quantenmechanik (nach [1], Abschnitt 2)

Gegeben sei ein Hamilton-Operator der Form:

$$H_1 = A^\dagger A \quad \text{mit} \quad A = \partial_x + W(x), A^\dagger = -\partial_x + W(x) \tag{1}$$

H_1 hat die Form $H_1 = -\partial_x^2 + V_1(x)$, wobei $V_1(x)$ gegeben ist durch:

$$V_1(x) = W^2(x) - W'(x) \tag{2}$$

$W(x)$ wird Superpotential genannt. Zu H_1 kann man nun einen supersymmetrischen Partner H_2 definieren:

$$H_2 = AA^\dagger \quad \Rightarrow \quad V_2(x) = W^2(x) + W'(x) \tag{3}$$

Die Eigenwerte von H_1 und H_2 müssen größer oder gleich null sein. Falls es einen (normierbaren) Zustand $\psi_0^{(1)}(x)$ bzw. $\psi_0^{(2)}(x)$ gibt mit $A\psi_0^{(1)}(x) = 0$ bzw. $A^\dagger\psi_0^{(2)}(x) = 0$, dann ist dieser also der Grundzustand von H_1 bzw. H_2 zum Eigenwert null. Dies führt auf:

$$A\psi_0^{(1)}(x) = 0 \quad \Rightarrow \quad W(x) = -\frac{\psi_0^{(1)\prime}(x)}{\psi_0^{(1)}(x)} = -\left[\log\psi_0^{(1)}(x)\right]' \tag{4}$$

$$A^\dagger\psi_0^{(2)}(x) = 0 \quad \Rightarrow \quad W(x) = +\left[\log\psi_0^{(1)}(x)\right]' \tag{5}$$

Oder:

$$\psi_0^{(1)}(x) = N \exp\left(-\int^x W(y)\mathrm{d}y\right) = \frac{1}{\psi_0^{(2)}(x)} \tag{6}$$

Im folgenden wird angenommen, dass $W(x)$ so beschaffen ist, dass $\psi_0^{(1)}(x)$ normierbar ist, H_1 soll also einen Eigenwert null besitzen. $\psi_0^{(2)}(x)$ ist dann nicht normierbar, d.h. H_2 hat keinen Energieeigenwert null.

Ist $\psi_n^{(1)}(x)$ mit $n > 0$ Eigenfunktion von H_1 zum Eigenwert $E_n^{(1)}$, dann ist $A\psi_n^{(1)}(x)$ Eigenfunktion von H_2 zum gleichen Eigenwert $E_n^{(1)}$:

$$H_2(A\psi_n^{(1)}(x)) = AA^\dagger A\psi_n^{(1)}(x) = AH_1\psi_n^{(1)}(x) = E_n^{(1)}(A\psi_n^{(1)}(x)) \tag{7}$$

Analog gilt auch: Ist $\psi_n^{(2)}(x)$ Eigenfunktion von H_2 zum Eigenwert $E_n^{(2)}$, dann ist $A^\dagger\psi_n^{(2)}(x)$ Eigenfunktion von H_1 zum gleichen Eigenwert $E_n^{(2)}$. H_1 und H_2 besitzen also bis auf den Grundzustand von H_1 das gleiche Spektrum. Der Grundzustand von H_2 hat die gleiche Energie wie der erste angeregte Zustand von H_1. Es gelten also folgende Beziehungen:

$$E_n^{(2)} = E_{n+1}^{(1)} \quad \text{mit} \quad E_0^{(1)} = 0 \tag{8}$$

$$\psi_n^{(2)}(x) = \left[E_{n+1}^{(1)}\right]^{-1/2} A\psi_{n+1}^{(1)}(x) \tag{9}$$

$$\psi_{n+1}^{(1)}(x) = \left[E_n^{(2)}\right]^{-1/2} A^\dagger\psi_n^{(2)}(x) \tag{10}$$

2.2 Isospektrale Potentiale (nach [1], Abschnitt 3 und 7)

Weil die Grundzustandsenergie nicht immer bei null liegt, ist ein allgemeinerer Ansatz für den Hamilton-Operator:

$$H_1 = A_1^\dagger A_1 + E_0^{(1)} \tag{11}$$

Analog zu oben gilt:

$$A_1 = \partial_x + W_1(x), \quad A_1^\dagger = -\partial_x + W_1(x), \quad W_1(x) = -\left[\log\psi_0^{(1)}(x)\right]' \tag{12}$$

Der supersymmetrische Partner lautet:

$$H_2 = A_1 A_1^\dagger + E_0^{(1)} \tag{13}$$

Das Potential $V_2(x)$ ist dann gegeben durch:

$$V_2(x) = W_1^2 + W_1' + E_0^{(1)} = V_1(x) + 2W_1' = V_1(x) - 2\frac{\mathrm{d}^2}{\mathrm{d}x^2}\log\psi_0^{(1)} \tag{14}$$

Wie in Abschnitt 2.1 erläutert, unterscheiden sich das Spektrum von $V_1(x)$ und $V_2(x)$ dadurch, dass der Grundzustand von $V_1(x)$ bei $V_2(x)$ entfernt wurde. $V_2(x)$ besitzt also einen Grundzustand $\psi_0^{(2)}(x)$ bei einer höheren Energie $E_0^{(2)}$. H_2 kann daher auch geschrieben werden als:

$$H_2 = A_2^\dagger A_2 + E_0^{(2)} \tag{15}$$

$$A_2 = \partial_x + W_2(x), \quad A_2^\dagger = -\partial_x + W_2(x), \quad W_2(x) = -\left[\log\psi_0^{(2)}\right]' \tag{16}$$

Der supersymmetrische Partner $H_3 = A_2 A_2^\dagger + E_0^{(2)}$ besitzt dann ein Potential $V_3(x)$ gegeben durch:

$$V_3(x) = V_2(x) - 2\frac{\mathrm{d}^2}{\mathrm{d}x^2}\log\psi_0^{(2)} = V_1(x) - 2\frac{\mathrm{d}^2}{\mathrm{d}x^2}\log\psi_0^{(1)}\psi_0^{(2)} \tag{17}$$

Beim Spektrum von $V_3(x)$ fehlen nun die zwei untersten Zustände von $V_1(x)$. Besitzt $V_1(x)$ n gebundene Zustände, dann kann man nun so fortfahren, bis man zum Potential $V_{n+1}(x)$ gelangt, das keine gebundenen Zustände mehr besitzt. Jetzt können zu den ursprünglichen Einergieeigenwerten von $V_1(x)$ neue Zustände zu $V_{n+1}(x)$ hinzugefügt werden, die aber jeweils von einem Parameter λ_i abhängen. Man erhält so eine n-parametrige Schar $\hat{V}_1(\lambda_1, ..., \lambda_n; x)$ von Potentialen, die das gleiche Spektrum wie $V_1(x)$ besitzen.

Dieses Verfahren soll am Beispiel $n = 2$ erläutert werden. Zunächst wird die Funktion $I_i(x)$ eingeführt:

$$I_i(x) = \int_{-\infty}^{x} \left[\psi_0^{(i)}\right]^2 \mathrm{d}x \tag{18}$$

Man sucht nun eine (nicht normierbare) Lösung der Schrödingergleichung zum Potential V_2 und zur Energie $E_0^{(1)}$. Nach (6) ist $1/\psi_0^{(1)}$ eine solche Lösung. Eine weitere linear unabhängige Lösung ist $I_1/\psi_0^{(1)}$, wie man durch einsetzen in die Schrödingergleichung feststellen kann[1]. Die allgemeine Lösung der Schrödingergleichung zum Potential V_2 bei der Energie $E_0^{(1)}$ kann also geschrieben werden als $\phi_1(\lambda_1)$ mit[2]:

$$\phi_i(\lambda_i) = \frac{I_i + \lambda_i}{\psi_0^{(i)}} \tag{19}$$

Ebenso ist $\phi_2(\lambda_2)$ Lösung der Schrödingergleichung zum Potential V_3 bei der Energie $E_0^{(2)}$, und $A_2\phi_1(\lambda_1)$ Lösung der Schrödingergleichung zum Potential V_3 bei der Energie $E_0^{(1)}$. Nun wird das Potential $\hat{V}_2(\lambda_2)$ gebildet:

$$\hat{V}_2(\lambda_2) = V_3 - 2\frac{\mathrm{d}^2}{\mathrm{d}x^2}\log\phi_2(\lambda_2) \tag{20}$$

Nach (6) ist nun $1/\phi_2(\lambda_2)$ Lösung der Schrödingergleichung zum Potential $\hat{V}_2(\lambda_2)$ bei der Energie $E_0^{(2)}$. Weil $1/\phi_2(\lambda_2)$ aber wiederum normierbar ist, besitzt $\hat{V}_2(\lambda_2)$ nun einen Grundzustand mit der Energie $E_0^{(2)}$. Weil V_3 keine gebundenen Zustände hatte, ist dieser Grundzustand auch der einzige gebundene Zustand von $\hat{V}_2(\lambda_2)$. Der Operator $\hat{A}_2^\dagger(\lambda_2)$, der Lösungen der Schrödingergleichung zum Potential V_3 auf Lösungen zum Potential $\hat{V}_2(\lambda_2)$ abbildet, ist gegeben durch:

$$\hat{A}_2^\dagger(\lambda_2) = -\partial_x + \hat{W}_2 \quad \text{mit} \quad \hat{W}_2 = +\left[\log\phi_2(\lambda_2)\right]' \tag{21}$$

Die Funktion $\phi_1(\lambda_1, \lambda_2)$ mit

$$\phi_1(\lambda_1, \lambda_2) = \hat{A}_2^\dagger(\lambda_2)A_2\phi_1(\lambda_1) \tag{22}$$

ist dann Lösung der Schrödingergleichung zum Potential $\hat{V}_2(\lambda_2)$ bei der Energie $E_0^{(1)}$. Das Potential $\hat{V}_1(\lambda_1, \lambda_2)$ ist nun analog zu (20):

$$\begin{aligned} \hat{V}_1(\lambda_1, \lambda_2) &= \hat{V}_2(\lambda_2) - 2\frac{\mathrm{d}^2}{\mathrm{d}x^2}\log\phi_1(\lambda_1, \lambda_2) \\ &= V_3 - 2\frac{\mathrm{d}^2}{\mathrm{d}x^2}\log\phi_2(\lambda_2)\phi_1(\lambda_1, \lambda_2) \end{aligned} \tag{23}$$

Weil $1/\phi_1(\lambda_1, \lambda_2)$ normierbar ist, hat $\hat{V}_1(\lambda_1, \lambda_2)$ einen Grundzustand mit Energie $E_0^{(1)}$. Außerdem ist $\hat{V}_2(\lambda_2)$ supersymmetrisches Partnerpotential von $\hat{V}_1(\lambda_1, \lambda_2)$, so dass der erste angeregte Zustand von $\hat{V}_1(\lambda_1, \lambda_2)$ die gleiche Energie wie der Grundzustand von $\hat{V}_2(\lambda_2)$ hat, nämlich $E_0^{(2)} = E_1^{(1)}$. V_1 und $\hat{V}_1(\lambda_1, \lambda_2)$ haben also das gleiche Spektrum.

Im Fall $n = 1$ lautet das Potential $\hat{V}_1(\lambda)$ einfach:

$$\hat{V}_1(\lambda) = V_2 - 2\frac{\mathrm{d}^2}{\mathrm{d}x^2}\log\phi_1(\lambda) \tag{24}$$

[1] Es gilt nämlich für $H = -\partial_x^2 + V(x)$: Wenn $H\Psi = E\Psi$, dann gilt auch $H\Phi = E\Phi$ mit $\Phi = \Psi\int_{-\infty}^{x}\Psi^{-2}\mathrm{d}x$.

[2] Man beachte, dass $1/\phi_i(\lambda_i)$ normierbar ist, falls $\lambda_i \notin [-1, 0]$.

3 Bestimmung der Parametertransformation

3.1 Das Potential $V(x) = -n(n+1)\operatorname{sech}^2 x$

Ausgangspunkt ist das folgende Superpotential:

$$W_1 = n \tanh x \quad \text{mit} \quad n \in \mathbb{N} \tag{25}$$

Der Hamilton-Operator H_1 soll gegeben sein durch (11) mit $E_0^{(1)} = -n^2$. Das Potential V_1 lautet dann:

$$\begin{aligned} V_1 &= W_1^2 - W_1' + E_0^{(1)} \\ &= n^2 \left(1 - \frac{1}{\cosh^2 x}\right) - \frac{n}{\cosh^2 x} - n^2 \\ &= -n(n+1)\operatorname{sech}^2 x \end{aligned} \tag{26}$$

Der Grundzustand kann mit (6) berechnet werden:

$$\begin{aligned} \psi_0^{(1)} &= N \exp\left(-n \log\cosh x\right) \\ &= N \operatorname{sech}^n x \end{aligned} \tag{27}$$

Das Potential V_2 lautet:

$$\begin{aligned} V_2 &= W_1^2 + W_1' + E_0^{(1)} \\ &= -n(n-1)\operatorname{sech}^2 x \end{aligned}$$

Man gelangt also vom Potential V_1 zum Potential V_2, indem man n durch $n-1$ ersetzt. Das bedeutet, dass man sofort das Superpotential W_2, den Grundzustand $\psi_0^{(2)}$ und seine Energie $E_0^{(2)}$ erhält, indem man in obigen Formeln n durch $n-1$ ersetzt. Entsprechend erhält man V_i, W_i, $\psi_0^{(i)}$ und $E_0^{(i)}$ indem man n durch $n-i+1$ ersetzt. Es gilt also:

$$W_i = (n-i+1)\tanh x \tag{28}$$

$$E_0^{(i)} = -(n-i+1)^2 \tag{29}$$

$$V_i = -(n-i+1)(n-i+2)\operatorname{sech}^2 x \tag{30}$$

$$\psi_0^{(i)} = N_i(\operatorname{sech} x)^{n-i+1} \tag{31}$$

Weil $V_{n+1} = 0$ keine gebundenen Zustände mehr besitzt, hat V_1 n gebundene Zustände. Die Energieniveaus $E_i^{(1)}$ und Eigenfunktionen $\psi_i^{(1)}$ können auf einfache Weise durch wiederholtes Anwenden der Relationen (8) und (10) aus $E_0^{(i)}$ und $\psi_0^{(i)}$ bestimmt werden[3]. V_1 ist ein sogenanntes „shape invariant potential“ (SIP). Für SIPs kann das hier skizzierte Lösungsverfahren für das Eigenwertproblem von H_1 allgemein formuliert werden (vgl. [1], Abschnitt 4).

Auf das Potential V_1 kann man nun das in Abschnitt 2.2 beschriebene Verfahren anwenden und erhält so ein Potential $\hat{V}_1(\lambda_1, ..., \lambda_n)$. Durch eine geeignete Parametertransformation zwischen den λ_i und der Zeit t wird $\hat{V}_1(\lambda_1(t), ..., \lambda_n(t))$ eine n-Solitonenlösung der KdV-Gleichung ([1], Abschnitt 7.3):

$$u_t + u_{xxx} - 6uu_x = 0 \tag{32}$$

3.2 Parametertransformation für $n = 1$

Das Potential (26) lautet für $n = 1$:

$$V_1 = -2\operatorname{sech}^2 x \tag{33}$$

[3] Dabei muss beachtet werden, dass bei den Relationen (8) und (10) ein Hamilton-Operator der Form $H_1 = A^\dagger A$ zugrunde gelegt wurde, und nicht $H_1 = A_1^\dagger A_1 + E_0^{(1)}$, die Formeln müssen also entsprechend angepasst werden.

Es besitzt einen gebundenen Zustand bei $E_0^{(1)} = -1$ mit Wellenfunktion $\psi_0^{(1)} = N \operatorname{sech} x$. Die Funktion I_1 ergibt sich aus (18) zu $I_1(x) = N^2(1 + \tanh x)$. Aus der Bedingung $I_1(\infty) = 1$ erhält man $N = 1/\sqrt{2}$. Weil $V_2 = 0$ ist, vereinfacht sich (24) zu:

$$\hat{V}_1(\lambda) = -2\frac{\mathrm{d}^2}{\mathrm{d}x^2} \log \phi_1(\lambda) \tag{34}$$

$\phi_1(\lambda)$ wird nach (19) berechnet:

$$\begin{aligned}
\phi_1(\lambda) &= \frac{\frac{1}{2}(1 + \tanh x) + \lambda}{\frac{1}{\sqrt{2}} \operatorname{sech} x} \\
&= \frac{\sqrt{2}}{2}(\cosh x + \sinh x + 2\lambda \cosh x) \\
&= \frac{\sqrt{2}}{2}\lambda e^{-x}\left[1 + \left(1 + \frac{1}{\lambda}\right) e^{2x}\right] \\
&= \frac{\sqrt{2}}{2}\lambda e^{-x}\left[1 + e^{2(x-\delta)}\right] \quad \text{mit} \quad \delta = -\frac{1}{2}\log\left(1 + \frac{1}{\lambda}\right)
\end{aligned} \tag{35}$$

Beachtet man, dass der Faktor $\frac{\sqrt{2}}{2}\lambda e^{-x}$ beim Ableiten in (34) herausfällt, dann erhält man:

$$\begin{aligned}
\hat{V}_1(x, \delta) &= -2\frac{\mathrm{d}^2}{\mathrm{d}x^2} \log\left[1 + e^{2(x-\delta)}\right] \\
&= -2\frac{\mathrm{d}}{\mathrm{d}x} \frac{2}{e^{-2(x-\delta)} + 1} \\
&= -2\frac{4e^{-2(x-\delta)}}{\left[e^{-2(x-\delta)} + 1\right]^2} \\
&= -2 \operatorname{sech}^2(x - \delta)
\end{aligned} \tag{36}$$

Nun lässt man δ von t abhängen und setzt (36) in die KdV-Gleichung (32) ein, d.h. $u(x,t) = \hat{V}_1(x, \delta(t))$:

$$u_x = \frac{4 \sinh(x - \delta)}{\cosh^3(x - \delta)} = -2u \tanh(x - \delta) \tag{37}$$

$$\begin{aligned}
u_{xx} &= -2\left[-2u \tanh(x - \delta)\right] \tanh(x - \delta) - 2u \operatorname{sech}^2(x - \delta) \\
&= 4u\left[1 - \operatorname{sech}^2(x - \delta)\right] - 2u \operatorname{sech}^2(x - \delta) \\
&= 4u + 3u^2
\end{aligned} \tag{38}$$

$$u_{xxx} = 4u_x + 6uu_x \tag{39}$$

$$u_t = -u_x \cdot \delta_t \tag{40}$$

$$\overset{(32)}{\Longrightarrow} \delta_t = -\frac{1}{u_x}(-u_{xxx} + 6uu_x) = 4 \tag{41}$$

Die gesuchte Parametertransformation ist also gegeben durch:

$$-\frac{1}{2}\log\left(1 + \frac{1}{\lambda}\right) = \delta = 4t + C \quad \text{mit} \quad C = \text{const} \tag{42}$$

C ist die Position des Solitons zur Zeit $t = 0$. Die Amplitude des Solitons kann man ändern, indem man folgende Substitution durchführt ($a = \text{const}$):

$$x \longrightarrow a \cdot x \tag{43}$$

$$t \longrightarrow a^3 \cdot t \tag{44}$$

$$u \longrightarrow a^2 \cdot u \tag{45}$$

Die KdV-Gleichung (32) bleibt invariant unter dieser Substitution. Die allgemeinste ein-Solitonenlösung ist also:

$$u = -2a^2 \operatorname{sech}^2(ax - 4a^3 t - ax_0) \tag{46}$$

Man sieht, dass die Geschwindigkeit des Solitons proportional zu seiner Amplitude ist.

3.3 Parametertransformation für $n = 2$

Das Potential (26) lautet für $n = 2$:

$$V_1 = -6 \operatorname{sech}^2 x \tag{47}$$

Es besitzt zwei gebundene Zustände bei $E_0^{(1)} = -4$ und $E_0^{(2)} = -1$. Die Wellenfunktionen der Grundzustände sind $\psi_0^{(1)} = N_1 \operatorname{sech}^2 x$ und $\psi_0^{(2)} = N_2 \operatorname{sech} x$. Die Funktion I_2 ergibt sich aus (18) zu $I_2(x) = N_2^2(1 + \tanh x)$. Die Bestimmung von I_1 gelingt mit Hilfe der Substitution $z = \tanh y$:

$$\begin{aligned} I_1(x) &= N_1^2 \int_{-\infty}^{x} \operatorname{sech}^4 y \, \mathrm{d}y \\ &= N_1^2 \int_{-\infty}^{x} \left(1 - \tanh^2 y\right) (\tanh y)' \, \mathrm{d}y \\ &= N_1^2 \int_{-1}^{\tanh x} \left(1 - z^2\right) \mathrm{d}z \\ &= \frac{N_1^2}{3} \left(3 \tanh x - \tanh^3 x + 2\right) \end{aligned} \tag{48}$$

Aus den Bedingungen $I_i(\infty) = 1$ erhält man $N_1 = \sqrt{3}/2$ und $N_2 = 1/\sqrt{2}$. Weil $V_3 = 0$ ist, vereinfacht sich (23) zu:

$$\hat{V}_1(\lambda_1, \lambda_2) = -2 \frac{\mathrm{d}^2}{\mathrm{d}x^2} \log \phi_2(\lambda_2) \phi_1(\lambda_1, \lambda_2) \tag{49}$$

Um die Parametertransformation für $n = 2$ zu finden, ist es nicht mehr zweckmäßig, $\hat{V}_1(\lambda_1, \lambda_2)$ in die KdV-Gleichung einzusetzen. In [3], Abschnitt 3.3.a ist folgende Formel für die n-Solitonenlösung der KdV-Gleichung angegeben[4]:

$$u = -2 \frac{\mathrm{d}^2}{\mathrm{d}x^2} \log F_n \tag{50}$$

Für F_n wird eine allgemeine Formel angegeben, die für $n = 1$ und $n = 2$ folgendermaßen lautet:

$$F_1 = 1 + e^{\eta_1} \tag{51}$$

$$F_2 = 1 + e^{\eta_1} + e^{\eta_2} + \left(\frac{k_1 - k_2}{k_1 + k_2}\right)^2 e^{\eta_1 + \eta_2} \tag{52}$$

$$\eta_i = k_i x - k_i^3 t + \eta_i^{(0)} \quad \text{mit} \quad k_i, \eta_i^{(0)} = \text{const} \tag{53}$$

Durch Vergleich von (50) mit (49) erkennt man, dass folgender Zusammenhang gelten muss:

$$\phi_2(\lambda_2) \phi_1(\lambda_1, \lambda_2) = \alpha_1 e^{\alpha_2 x} \cdot F_2 \quad \text{mit} \quad \alpha_1, \alpha_2 = \text{const} \tag{54}$$

Der Faktor $\alpha_1 e^{\alpha_2 x}$ fällt nämlich beim Ableiten in (49) heraus. An Gleichung (35) erkennt man, dass für den Fall $n = 1$ ein analoger Zusammenhang gegeben ist. Man hätte so durch Vergleich von (35) mit (51) bereits die Parametertransformation für $n = 1$ ablesen können. Dieses Verfahren soll hier für den Fall $n = 2$ verwendet werden.

[4] In [3] wird die KdV-Gleichung in der Form $u_t + u_{xxx} + 6uu_x = 0$ verwendet, so dass u durch $-u$ ersetzt werden muss.

Es folgt die Bestimmung von $\phi_2(\lambda_2)\phi_1(\lambda_1,\lambda_2)$. Es ist dazu zweckmäßig, δ_i wie folgt zu definieren:

$$e^{-2\delta_i} = 1 + \frac{1}{\lambda_i} \quad \text{bzw.} \quad \delta_i = -\frac{1}{2}\log\left(1+\frac{1}{\lambda_i}\right) \tag{55}$$

Zunächst wird $\phi_1(\lambda_1)$ nach (19) berechnet:

$$\begin{aligned}\phi_1(\lambda_1) &= \frac{\frac{1}{4}\left(3\tanh x - \tanh^3 x + 2\right) + \lambda_1}{\frac{\sqrt{3}}{2}\operatorname{sech}^2 x} \\ &= \frac{1}{2\sqrt{3}}\left[3\sinh x\cosh x - \sinh^2 x\tanh x + 2(1+2\lambda_1)\cosh^2 x\right] \end{aligned}\tag{56}$$

$\phi_2(\lambda_2)$ wurde bereits in (35) berechnet:

$$\phi_2(\lambda_2) = \frac{\sqrt{2}}{2}\lambda_2\left(e^{-x} + e^{x-2\delta_2}\right) \tag{57}$$

Für $\phi_1(\lambda_1,\lambda_2)$ wird nach (22) $A_2\phi_1(\lambda_1)$ benötigt. Dabei ist $A_2 = \partial_x + W_2$ mit $W_2 = \tanh x$:

$$\partial_x\phi_1(\lambda_1) = \frac{1}{2\sqrt{3}}\left[3\cosh^2 x + \sinh^2 x - \tanh^2 x + 4(1+2\lambda_1)\cosh x\sinh x\right] \tag{58}$$

$$\tanh x\cdot\phi_1(\lambda_1) = \frac{1}{2\sqrt{3}}\left[3\sinh^2 x - \sinh^2 x\tanh^2 x + 2(1+2\lambda_1)\cosh x\sinh x\right] \tag{59}$$

$$\begin{aligned}\Longrightarrow A_2\phi_1(\lambda_1) &= \frac{3}{2\sqrt{3}}\left[\cosh^2 x + \sinh^2 x + 2(1+2\lambda_1)\cosh x\sinh x\right] \\ &= \frac{\sqrt{3}}{2}\left[\frac{1}{2}\left(e^{2x}+e^{-2x}\right) + 2(1+2\lambda_1)\frac{1}{4}\left(e^{2x}-e^{-2x}\right)\right] \\ &= \frac{\sqrt{3}}{2}\lambda_1\left(e^{2x-2\delta_1} - e^{-2x}\right)\end{aligned}\tag{60}$$

Man erhält schließlich für $\phi_2(\lambda_2)\phi_1(\lambda_1,\lambda_2)$ mit (22), (21), (60) und (57):

$$\begin{aligned}\phi_2(\lambda_2)\phi_1(\lambda_1,\lambda_2) &= \phi_2(\lambda_2)\left[-\partial_x + [\log\phi_2(\lambda_2)]'\right]A_2\phi_1(\lambda_1) \\ &= \left[-\phi_2(\lambda_2)\partial_x + \phi_2'(\lambda_2)\right]A_2\phi_1(\lambda_1) \\ \phi_2(\lambda_2)\partial_x A_2\phi_1(\lambda_1) &= \frac{\sqrt{6}}{4}\lambda_1\lambda_2\left(e^{-x}+e^{x-2\delta_2}\right)\left(2e^{2x-2\delta_1}+2e^{-2x}\right) \\ &= \frac{\sqrt{6}}{4}\lambda_1\lambda_2\left(2e^{-3x}+2e^{-x-2\delta_2}+2e^{x-2\delta_1}+2e^{3x-2\delta_1-2\delta_2}\right) \end{aligned}\tag{61}$$

$$\begin{aligned}\phi_2'(\lambda_2)A_2\phi_1(\lambda_1) &= \frac{\sqrt{6}}{4}\lambda_1\lambda_2\left(-e^{-x}+e^{x-2\delta_2}\right)\left(e^{2x-2\delta_1}-e^{-2x}\right) \\ &= \frac{\sqrt{6}}{4}\lambda_1\lambda_2\left(e^{-3x}-e^{-x-2\delta_2}-e^{x-2\delta_1}+e^{3x-2\delta_1-2\delta_2}\right)\end{aligned}\tag{62}$$

$$\Longrightarrow \phi_2(\lambda_2)\phi_1(\lambda_1,\lambda_2) = -\frac{\sqrt{6}}{4}\lambda_1\lambda_2 e^{-3x}\left(1+3e^{2x-2\delta_2}+3e^{4x-2\delta_1}+e^{6x-2\delta_1-2\delta_2}\right) \tag{63}$$

Ein Vergleich von (63) mit (54) und (52) ergibt:

$$k_1 = 4, \qquad k_2 = 2 \tag{64}$$

$$-2\delta_1 + \log 3 = -4^3 t + \eta_1^{(0)} \tag{65}$$

$$-2\delta_2 + \log 3 = -2^3 t + \eta_2^{(0)} \tag{66}$$

Die gesuchte Parametertransformation ist also gegeben durch (55) und:

$$\delta_1 = 32t + C_1, \quad \delta_2 = 4t + C_2 \quad \text{mit} \quad C_1, C_2 = \text{const} \tag{67}$$

$\hat{V}_1(\delta_1, \delta_2)$ kann nun nach (49) berechnet werden:

$$\begin{aligned}
\hat{V}_1(\delta_1, \delta_2; x) &= -2\frac{\mathrm{d}^2}{\mathrm{d}x^2} \log\left(1 + 3e^{2x-2\delta_2} + 3e^{4x-2\delta_1} + e^{6x-2\delta_1-2\delta_2}\right) \\
&= -12\frac{\mathrm{d}}{\mathrm{d}x} \frac{e^{2x-2\delta_2} + 2e^{4x-2\delta_1} + e^{6x-2\delta_1-2\delta_2}}{1 + 3e^{2x-2\delta_2} + 3e^{4x-2\delta_1} + e^{6x-2\delta_1-2\delta_2}} \\
&= -24\frac{e^{2x-2\delta_2} + 4e^{4x-2\delta_1} + 6e^{6x-2\delta_1-2\delta_2} + 4e^{8x-2\delta_1-4\delta_2} + e^{10x-4\delta_1-2\delta_2}}{\left(1 + 3e^{2x-2\delta_2} + 3e^{4x-2\delta_1} + e^{6x-2\delta_1-2\delta_2}\right)^2} \\
&= -24\frac{e^{-4x+2\delta_1} + 4e^{-2x+2\delta_2} + 6 + 4e^{2x-2\delta_2} + e^{4x-2\delta_1}}{\left(e^{-3x+\delta_1+\delta_2} + 3e^{-x+\delta_1-\delta_2} + 3e^{x-\delta_1+\delta_2} + e^{3x-\delta_1-\delta_2}\right)^2} \\
&= -12\frac{3 + 4\cosh(2x - 2\delta_2) + \cosh(4x - 2\delta_1)}{\left[\cosh(3x - \delta_2 - \delta_1) + 3\cosh(x + \delta_2 - \delta_1)\right]^2} \qquad (68)
\end{aligned}$$

Dies stellt eine relativ einfache zwei-Solitonenlösung der KdV-Gleichung dar. Das Verhältnis der Amplituden der beiden Solitonen ist allerdings festgelegt, durch die Substitution (45) werden immer beide Amplituden zugleich verändert. Die Konstanten C_1 und C_2 legen die Startpositionen der Solitonen fest.

4 Numerische Simulation

Zur numerischen Lösung der KdV-Gleichung[5]

$$u_t + u_{xxx} + 6uu_x = 0 \qquad (69)$$

wurde zunächst der Raum in Abständen Δx und die Zeit in Abständen Δt diskretisiert. Die Lösung erfolgte dann nach folgendem Schema:

$$\begin{aligned}
u(x_i, t_{j+1}) &= u(x_i, t_j) + u_t(x_i, t_j) \cdot \Delta t \\
&= u(x_i, t_j) - [u_{xxx}(x_i, t_j) + 6u(x_i, t_j)u_x(x_i, t_j)] \cdot \Delta t \qquad (70)
\end{aligned}$$

Die einfachste Möglichkeit, $u_x(x_i, t_j)$ zu berechnen ist:

$$u_x(x_i, t_j) = \frac{1}{2\Delta x}\left[u(x_{i+1}, t_j) - u(x_{i-1}, t_j)\right] \qquad (71)$$

Höhere Ableitungen können durch mehrfaches Anwenden von (71) erhalten werden:

$$\begin{aligned}
u_{xx}(x_i, t_j) &= \frac{1}{2\Delta x}\left[u_x(x_{i+1}, t_j) - u_x(x_{i-1}, t_j)\right] \\
&= \frac{1}{4(\Delta x)^2}\left[u(x_{i+2}, t_j) - 2u(x_i, t_j) + u(x_{i-2}, t_j)\right] \qquad (72) \\
u_{xxx}(x_i, t_j) &= \frac{1}{2\Delta x}\left[u_{xx}(x_{i+1}, t_j) - u_{xx}(x_{i-1}, t_j)\right] \\
&= \frac{1}{8(\Delta x)^3}\left[u(x_{i+3}, t_j) - 3u(x_{i+1}, t_j) + 3u(x_{i-1}, t_j) - u(x_{i-3}, t_j)\right] \qquad (73)
\end{aligned}$$

Es zeigte sich jedoch, dass (71) nicht genau genug ist um $u_x(x_i, t_j)$ zu bestimmen. Daher wurde zunächst ein Polynom vierten Grades $a(x - x_i)^4 + b(x - x_i)^3 + c(x - x_i)^2 + d(x - x_i) + e$ durch die Punkte um x_i gelegt. $u_x(x_i, t_j)$ ist dann identisch mit dem Koeffizienten d. Mit $z = x - x_i$,

[5] Im Vergleich zu (32) wurde u durch $-u$ ersetzt.

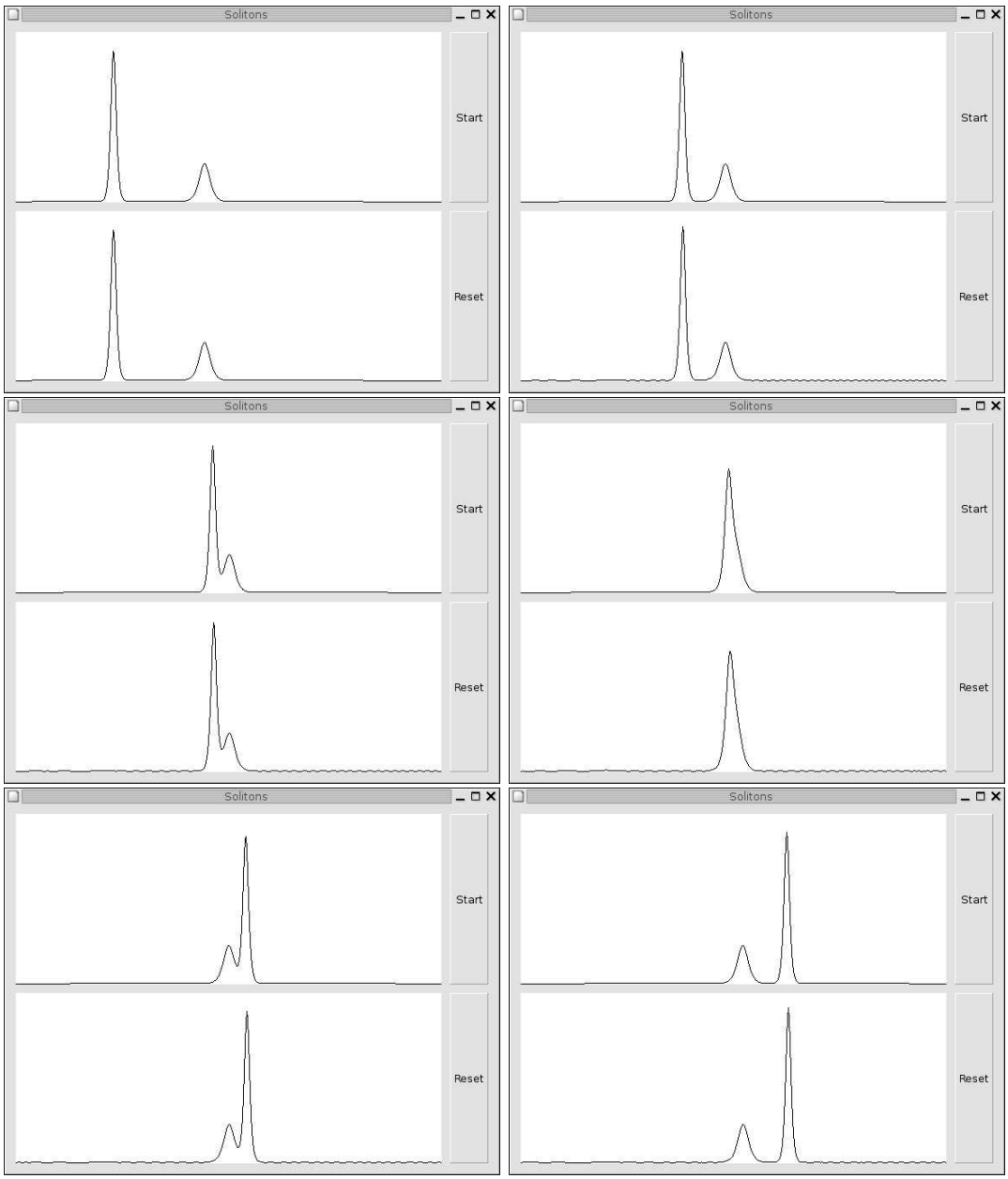

Abbildung 1: Screenshots des Programms in chronologischer Reihenfolge

$z_k = x_{i+k} - x_i$ und $y_k = u(x_{i+k}, t_j)$ lautet das Polynom:

$$\begin{aligned} & \frac{(z_{-1}-z)(z_0-z)(z_1-z)(z_2-z)}{(z_{-1}-z_{-2})(z_0-z_{-2})(z_1-z_{-2})(z_2-z_{-2})} & y_{-2} \\ + & \frac{(z_{-2}-z)(z_0-z)(z_1-z)(z_2-z)}{(z_{-2}-z_{-1})(z_0-z_{-1})(z_1-z_{-1})(z_2-z_{-1})} & y_{-1} \\ + & \frac{(z_{-2}-z)(z_{-1}-z)(z_1-z)(z_2-z)}{(z_{-2}-z_0)(z_{-1}-z_0)(z_1-z_0)(z_2-z_0)} & y_0 \\ + & \frac{(z_{-2}-z)(z_{-1}-z)(z_0-z)(z_2-z)}{(z_{-2}-z_1)(z_{-1}-z_1)(z_0-z_1)(z_2-z_1)} & y_1 \\ + & \frac{(z_{-2}-z)(z_{-1}-z)(z_0-z)(z_1-z)}{(z_{-2}-z_2)(z_{-1}-z_2)(z_0-z_2)(z_1-z_2)} & y_2 \end{aligned} \tag{74}$$

Dieses Polynom hat offensichtlich den Grad vier und stimmt an den Punkten z_k mit $k = -2, -1, 0, 1, 2$ mit $u(x_{i+k}, t_j)$ überein. Mit $z_k = k \cdot \Delta x$ erhält man für den Koeffizienten d:

$$d = \frac{1}{12\Delta x} \left[y_{-2} - y_2 + 8 \left(y_1 - y_{-1} \right) \right] \tag{75}$$

Eine bessere Formel für $u_x(x_i, t_j)$ ist also:

$$u_x(x_i, t_j) = \frac{1}{12\Delta x} \left[u(x_{i-2}, t_j) - u(x_{i+2}, t_j) + 8 \left[u(x_{i+1}, t_j) - u(x_{i-1}, t_j) \right] \right] \tag{76}$$

In Anhang A.1 ist ein Programm aufgelistet, welches die KdV-Gleichung mit Hilfe von (70), (73) und (76) numerisch löst. Das Programm benutzt libgtk2.0 zur Anzeige. Im oberen Fenster wird die analytische Lösung nach (68) und (67) angezeigt, im unteren Fenster die numerische Lösung. Die x-Achse erstreckt sich von -30 bis $+30$, die y-Achse von 0 bis 9. Die Startzeit ist bei $t = -1$, bei $t = 1.8$ erreicht das größere Soliton den rechten Rand und die Simulation endet. Abbildung 1 zeigt die Ausgabe des Programms.

A Quellcode

A.1 soli.c

```
/* soli.c - Simulation von zwei Solitonen
 *
 * Im oberen Fenster wird die analytische Loesung angezeigt,
 * im unteren die numerische.
 */

#include <gtk/gtk.h>
#include <math.h>
#include <pthread.h>
#include <semaphore.h>

/* x_start und x_end legen den x-Bereich fest, der
 * angezeigt werden soll. Entsprechend legen t_start
 * und t_end den Zeitbereich fest, der simuliert
 * werden soll. t_step gibt an, in welchen Abstaenden
 * der Bildschirm aktualisiert werden soll.
 */
double x_start = -30.0;
double x_end = 30.0;
double t_start = -1.0;
double t_end = 1.8;
double t_step = 0.002;
```

```
double t;

GdkPixmap *pix_numeric = NULL;
GdkPixmap *pix_analytic = NULL;
GtkWidget *draw1 = NULL;
int width1, height1;
GtkWidget *draw2 = NULL;
int width2, height2;
sem_t draw_config_sem;

/* die analytische Loesung */
double analytic(double x, double t)
{
        double tmp;

        tmp = cosh(3.0*x-((8.0+4.0*4.0*4.0)/2.0)*t)
              + 3.0*cosh(x+((8.0-4.0*4.0*4.0)/2.0)*t);

        return 12.0 * (3.0 + 4.0*cosh(2.0*x-8.0*t)
                       + cosh(4.0*x-(4.0*4.0*4.0)*t)
                      ) / (tmp*tmp);
}

/* zeichnen der analytischen Loesung in pix_analytic */
void draw_analytic()
{
        int i, y, last_y;
        double x, x_step, y_scale;

        y_scale = height1/9.0;
        x_step = (x_end-x_start)/width1;
        x = x_start;
        last_y = height1 - analytic(x, t)*y_scale - 1;

        gdk_draw_rectangle(pix_analytic,
                           draw1->style->white_gc,
                           TRUE, 0, 0, width1, height1);

        for (i=0; i<width1; i++) {
                x += x_step;
                y = height1 - analytic(x, t)*y_scale - 1;
                gdk_draw_line(pix_analytic,
                              draw1->style->black_gc,
                              i, last_y, i+1, y);
                last_y = y;
        }
}

/* Variablen fuer die numerische Berechnung */
#define xsteps 1000              /* Zahl der Punkte in x-Richtung */
double numeric_xstep;            /* Schrittweite in x-Richtung */
#define numeric_tstep 0.0000005 /* Schrittweite in t-Richtung */
double numeric_data1[xsteps];
double numeric_data2[xsteps];
```

```
double *numeric_data = numeric_data1;
double *last_numeric_data = numeric_data2;
double last_t;

/* Festlegen der Anfangswerte fuer die numerische Loesung */
void reset_numeric()
{
        int i;

        t = t_start;
        last_t = t_start;

        numeric_xstep = (x_end-x_start)/(double)xsteps;

        for (i=0; i<xsteps; i++) {
                numeric_data[i] = analytic(x_start
                                        + numeric_xstep*i,
                                        t);
                last_numeric_data[i] = numeric_data[i];
        }
}

/* numerische Berechnung der ersten Ableitung nach x */
inline double calc_d(int i)
{
        return (last_numeric_data[i-2]
                - last_numeric_data[i+2]
                + 8.0*(last_numeric_data[i+1]
                        - last_numeric_data[i-1])
               ) / (12.0*numeric_xstep);
}

/* numerische Berechnung der dritten Ableitung nach x */
inline double calc_d3(int i)
{
        return (last_numeric_data[i+3]
                - 3.0*last_numeric_data[i+1]
                + 3.0*last_numeric_data[i-1]
                - last_numeric_data[i-3]
               ) / (8.0*numeric_xstep
                    *numeric_xstep*numeric_xstep);
}

/* numerische Berechnung der Loesung */
void calc_numeric()
{
        int i;
        double dudt;
        double *tmp;

        while ((t-last_t)<t_step){
                t += numeric_tstep;
                tmp = numeric_data;
                numeric_data = last_numeric_data;
```

```
                last_numeric_data = tmp;

                for (i=3; i<xsteps-3; i++) {
                        dudt = - calc_d3(i)
                               - 6.0*last_numeric_data[i]*calc_d(i);

                        numeric_data[i] = last_numeric_data[i]
                                        + dudt*numeric_tstep;
                }

                pthread_testcancel();
        }

        last_t = t;
}

/* zeichnen der numerischen Loesung in pix_numeric */
void draw_numeric()
{
        int i, y, last_y, xi;
        double x, x_step, y_scale;

        y_scale = height2/9.0;
        x_step = (x_end-x_start)/width2;
        x = x_start;
        last_y = height2 - analytic(x, t)*y_scale - 1;

        gdk_draw_rectangle(pix_numeric,
                           draw2->style->white_gc,
                           TRUE, 0, 0, width2, height2);

        for (i=0; i<width2; i++) {
                x += x_step;
                xi = (x-x_start)/numeric_xstep;
                y = height2 - numeric_data[xi]*y_scale - 1;
                gdk_draw_line(pix_numeric,
                              draw2->style->black_gc,
                              i, last_y, i+1, y);
                last_y = y;
        }
}

/* aktualisieren der beiden Graphen */
void update()
{
        gdk_draw_drawable(draw1->window,
                          draw1->style
                               ->fg_gc[GTK_WIDGET_STATE(draw1)],
                          pix_analytic, 0, 0, 0, 0,
                          width1, height1);
        gdk_draw_drawable(draw2->window,
                          draw2->style
                               ->fg_gc[GTK_WIDGET_STATE(draw2)],
                          pix_numeric, 0, 0, 0, 0,
```

```
                                width2, height2);
}

/* die Hauptschleife, wird bei druecken
 * des Start-buttons ausgefuehrt */
void *run(void *arg)
{
        while(t<t_end) {
                calc_numeric();

                sem_wait(&draw_config_sem);
                draw_numeric();
                draw_analytic();
                update();
                sem_post(&draw_config_sem);
        }

        return NULL;
}

/* es folgen einige Funktionen zum Abhandeln der events */

void destroy(GtkWidget *widget, gpointer data)
{
        gtk_main_quit();
}

gboolean configure_event1(GtkWidget *widget, GdkEventConfigure *event)
{
        sem_wait(&draw_config_sem);

        if (pix_analytic)
                g_object_unref(pix_analytic);

        width1 = draw1->allocation.width;
        height1 = draw1->allocation.height;
        pix_analytic = gdk_pixmap_new(widget->window,
                                      width1, height1, -1);
        draw_analytic();

        sem_post(&draw_config_sem);

        return TRUE;
}

gboolean configure_event2(GtkWidget *widget, GdkEventConfigure *event)
{
        sem_wait(&draw_config_sem);

        if (pix_numeric)
                g_object_unref(pix_numeric);

        width2 = draw2->allocation.width;
        height2 = draw2->allocation.height;
```

```
        pix_numeric = gdk_pixmap_new(widget->window,
                                     width2, height2, -1);
        draw_numeric();

        sem_post(&draw_config_sem);

        return TRUE;
}

gboolean expose_event1(GtkWidget *widget,
                       GdkEventExpose *event,
                       GdkPixmap **pixmap)
{
        gdk_draw_drawable(widget->window,
                          widget->style
                                ->fg_gc[GTK_WIDGET_STATE(widget)],
                          pix_analytic,
                          event->area.x, event->area.y,
                          event->area.x, event->area.y,
                          event->area.width, event->area.height);

        return FALSE;
}

gboolean expose_event2(GtkWidget *widget,
                       GdkEventExpose *event,
                       GdkPixmap **pixmap)
{
        gdk_draw_drawable(widget->window,
                          widget->style
                                ->fg_gc[GTK_WIDGET_STATE(widget)],
                          pix_numeric,
                          event->area.x, event->area.y,
                          event->area.x, event->area.y,
                          event->area.width, event->area.height);

        return FALSE;
}

pthread_t tr;
int running = 0;

void start(GtkWidget *widget, gpointer data)
{
        if (running)
                return;

        pthread_create(&tr, NULL, run, NULL);
        running = 1;
}

void reset(GtkWidget *widget, gpointer data)
{
        if (!running)
```

```
                return;

        pthread_cancel(tr);
        running = 0;
        reset_numeric();
        draw_numeric();
        draw_analytic();
        update();
}

/* stetzen der Anfangswerte und Aufbau des Fensters */
int main(int argc, char *argv[])
{
        GtkWidget *window, *button1, *button2, *box1, *box2, *box3;

        sem_init(&draw_config_sem, 0, 1);
        reset_numeric();
        gtk_init(&argc, &argv);

        window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
        gtk_window_set_title(GTK_WINDOW(window), "Solitons");
        g_signal_connect(G_OBJECT(window), "destroy",
                         G_CALLBACK(destroy), NULL);
        gtk_container_set_border_width(GTK_CONTAINER(window), 10);

        box1 = gtk_hbox_new(FALSE, 10);
        gtk_container_add(GTK_CONTAINER(window), box1);
        gtk_widget_show(box1);

        box2 = gtk_vbox_new(FALSE, 10);
        gtk_box_pack_start(GTK_BOX(box1), box2, TRUE, TRUE, 0);
        gtk_widget_show(box2);

        box3 = gtk_vbox_new(FALSE, 10);
        gtk_box_pack_start(GTK_BOX(box1), box3, TRUE, TRUE, 0);
        gtk_widget_show(box3);

        draw1 = gtk_drawing_area_new();
        gtk_widget_set_size_request(draw1, 500, 200);
        gtk_signal_connect(GTK_OBJECT(draw1), "expose_event",
                           (GtkSignalFunc) expose_event1, NULL);
        gtk_signal_connect(GTK_OBJECT(draw1), "configure_event",
                           (GtkSignalFunc) configure_event1, NULL);
        gtk_box_pack_start(GTK_BOX(box2), draw1, TRUE, TRUE, 0);
        gtk_widget_show(draw1);

        draw2 = gtk_drawing_area_new();
        gtk_widget_set_size_request(draw2, 500, 200);
        gtk_signal_connect(GTK_OBJECT(draw2), "expose_event",
                           (GtkSignalFunc) expose_event2, NULL);
        gtk_signal_connect(GTK_OBJECT(draw2), "configure_event",
                           (GtkSignalFunc) configure_event2, NULL);
        gtk_box_pack_start(GTK_BOX(box2), draw2, TRUE, TRUE, 0);
        gtk_widget_show(draw2);
```

```
        button1 = gtk_button_new_with_label("Start");
        g_signal_connect(G_OBJECT(button1), "clicked",
                         G_CALLBACK(start), NULL);
        gtk_box_pack_start(GTK_BOX(box3), button1, TRUE, TRUE, 0);
        gtk_widget_show(button1);

        button2 = gtk_button_new_with_label("Reset");
        g_signal_connect(G_OBJECT(button2), "clicked",
                         G_CALLBACK(reset), NULL);
        gtk_box_pack_start(GTK_BOX(box3), button2, TRUE, TRUE, 0);
        gtk_widget_show(button2);

        gtk_widget_show(window);
        gtk_main();

        return 0;
}
```

A.2 Makefile

```
CFLAGS = -O3 -pipe -Wall `pkg-config --cflags gtk+-2.0`
CXXFLAGS = $(CFLAGS)
LDFLAGS = `pkg-config --libs gtk+-2.0` -lpthread

soli:
.PHONY: clean
clean:
        $(RM) -f *.o soli
```

Literatur

[1] F. Cooper, A. Khare, U. Sukhatme, Supersymmetry and Quantum Mechanics, arXiv:hep-th/9405029 v2, 1994

[2] R. Meinel, G.Neugebauer, H. Streudel, Solitonen - Nichtlineare Strukturen, Akademie Verlag, 1991

[3] M. J. Ablowitz, H. Segur, Solitons and the Inverse Scattering Transform, SIAM, 1985

[4] S. Novikov, S. V. Manakov, L. P. Pitaevskii, V. E. Zakharov, Theory of Solitons - The Inverse Scattering Method, Consultants Bureau, 1984