Energiespektrum aus Autokorrelationsfunktion

Pavel Sterin

1 Split-Operator Methode

Die Split-Operator Methode ist ein Verfahren zur numerischen Approximation von Lösungen der zeitabhängigen Schrödingergleichung in kartesichen Koordinaten. Zur Vereinfachung der Berechnungen verwende ich die Konvention $\hbar = 1$, $m = \frac{1}{2}$. Damit erhält man eine einfache Form, der linearen DGL:

$$\left(-\frac{\partial^2}{\partial x^2} + V(x,t)\right)\psi(x,t) = i\frac{\partial}{\partial t}\psi(x,t)$$
$$= (T+V)\,\psi(x,t) = i\frac{\partial}{\partial t}\psi(x,t)$$

Mit dem üblichen Ansatz für die zeitliche Evaluation von $\psi(x,t) = U(t)\psi(x)$ (mit fixierter Wellenfunktion $\psi(x)$ und unitärem Zeit-Propagator $U(t)$) erhält man die äquivalente Schrödingergleichung des Propagators:

$$(T + V(t))\,U(t) = i\frac{\partial}{\partial t}U(t)$$

Die formale Lösung dieser Gleichung ist bekannterweise die Dyson-Reihe, oder etwas weniger allgemein, für verschiedene Zeiten mit sich selbst kommutierenden Hammiltonian $H(t) = T + V(t)$, $[H(t), H(t')] = 0$ ist:

$$U(t) = \exp\left(\int_0^t -i(T + V(t'))\mathrm{d}t'\right)$$

Falls sich das Potential für kleine Zeitdiffrenzen τ nur geringfügig ändert oder sogar stets konstant ist, sodass $[H(t), H(t+\tau)] \approx 0$ gilt, geht diese Lösung in das Operator-Exponential über:

$$U(t,\tau) = \exp\left(-i\tau(T + V(t))\right)$$

Im weiteren beziehe ich mich stets auf eine einzige Iteration und lasse darum die zetliche Abhängikeit von V fallen:

$$U(\tau) = \mathrm{e}^{-i\tau(T+V)} \tag{1}$$

Da T nur im Impulsraum und V nur im Ortsraum wirken, wäre es günstig sie nur dort auszuwerten, weil sie dann als Multiplikationsopretoren diagonal wären und das Exponential sich leicht auswerten ließe. Da jedoch für die meisten interessanten Probleme T und V nicht Kommutieren, $[T,V] \neq 0$, muss man hier geschickt Approxomieren.

Für jeden Operator gilt zunächst:

$$\mathrm{e}^{-i\tau X} = \sum_{n=0}^{\infty}\frac{(i\tau X)^n}{n!} = 1 - i\tau X - \tau^2\frac{X^2}{2} + O(\tau^3)$$

Mit den Bezeichnung $U_X(\tau) = \mathrm{e}^{-i\tau X}$ ergibt das dann:

$$U_{X+Y}(\tau) = 1 - i\tau(X+Y) - \tau^2\frac{(X+Y)^2}{2} + O(\tau^3)$$
$$= 1 - i\tau X - i\tau Y$$
$$- \frac{\tau^2}{2}\left(X^2 + XY + YX + Y^2\right) + O(\tau^3)$$

$$U_X(\tau) = 1 - i\tau X - \tau^2\frac{X^2}{2} + O(\tau^3)$$
$$U_Y(\tau) = 1 - i\tau Y - \tau^2\frac{Y^2}{2} + O(\tau^3)$$

$$U_X(\tau)U_Y(\tau) = 1 - i\tau Y - i\tau X$$
$$- \tau^2 XY - \tau^2\frac{Y^2}{2} - \tau^2\frac{X^2}{2} + O(\tau^3)$$

Damit erhält man den Ausdruck:

$$U_{X+Y}(\tau) = U_X(\tau)U_Y(\tau) + \frac{\tau^2}{2}[X,Y] + O(\tau^3) \tag{2}$$

Mit $X = \frac{V}{2}$, $Y = T + \frac{V}{2}$ folgt:

$$U_{\frac{V}{2}+T+\frac{V}{2}}(\tau) = U_{\frac{V}{2}}(\tau)U_{T+\frac{V}{2}}(\tau)$$
$$+ \left[\frac{V}{2}, T + \frac{V}{2}\right] + O(\tau^3)$$
$$= U_{\frac{V}{2}}(\tau)U_{T+\frac{V}{2}}(\tau) + \left[\frac{V}{2}, T\right] + O(\tau^3)$$
$$= U_{\frac{V}{2}}(\tau)\left(U_T(\tau)U_{\frac{V}{2}}(\tau) + \frac{\tau^2}{2}\left[T, \frac{V}{2}\right]\right)$$
$$+ \left[\frac{V}{2}, T\right] + O(\tau^3)$$
$$= U_{\frac{V}{2}}(\tau)U_T(\tau)U_{\frac{V}{2}}(\tau) + O(\tau^3) \tag{3}$$

Durch die symmterische Aufteilung des Potentials verringert sich also der Fehler auf die 3. Ordnung.

In dieser Darstellung ist jetzt einfach die Wirkung von $U \approx \tilde{U} = U_{\frac{V}{2}}(\tau)U_T(\tau)U_{\frac{V}{2}}(\tau)$ zu berechnen, denn:

$$U_{\frac{V}{2}}(\tau)\psi(x) = \mathrm{e}^{-i\frac{\tau}{2}V(x)}\psi(x) \tag{4}$$

$$U_T(\tau)\psi(x) = \mathcal{F}^*\mathrm{e}^{-i\tau k^2}\mathcal{F}\psi(x) \tag{5}$$

wobei $\mathcal{F}$ die Fouriertransformation ist.

Diese beiden Gleichungen definieren die Iteration der Split-Operator-Methode für n Schtitte bzw. Zeit $t = n\tau$

Start
- Propagation von $\psi(x)$ mit $U_{\frac{V}{2}}(\tau)$
- Fouriertransformation zu $\psi(k)$
- Propagation von $\psi(k)$ mit $U_T(\tau)$

Iteration
- inverse Fouriertransformation zu $\psi(x)$
- Propagation von $\psi(x)$ mit $U_{\frac{V}{2}}(\tau)$
- Propagation von $\psi(x)$ mit $U_{\frac{V}{2}}(\tau)$
- Fouriertransformation zu $\psi(k)$
- Propagation von $\psi(k)$ mit $U_T(\tau)$

Ende
- inverse Fouriertransformation zu $\psi(x)$
- Propagation von $\psi(x)$ mit $U_{\frac{V}{2}}(\tau)$

wobei die Iteration $(n-1)$ Mal ausgeführt wird. Die zwei nacheinander folgenden Propagationen mit $U_{\frac{V}{2}}(\tau)$ werden in der Implementierung natürlich zu $U_V(\tau)$ zusammengefasst.

Bemerkung: Es hindert einen niemend daran die Rollen von T und V zu vertauschen und statt $\tilde{U}$ den Operator $U_{\frac{T}{2}}(\tau)U_V(\tau)U_{\frac{T}{2}}(\tau)$ zu verwenden. Der einzige Unterschied ist eine zusätliche Fouriertransformation am Anfang und eine inverse Fouriertransformation am Ende des Algorithmus, die mir angesichts der eher kurzen Iterationen für dieses Projekt unangebracht erschienen.

2 Autokorrelationsfunktion

Ein zeitunabhängiger Hamiltonian H eines Systems besitzt eine Basis in der er als Multiplikationsoperator wirkt. In dieser Basis besitzt H eine Spektralzerlegung der Form

$$H = \int_{\sigma(H)} \epsilon\mu(\mathrm{d}\epsilon)$$

mit Spektralmaß $\mu(\epsilon)$ und Spektrum $\sigma(H)$. Da H nun ein Multiplikationsoperator ist, lässt sich der Zeit-Propagator formal einfach auswerten zu:

$$U(t) = \exp\left(-it\int_{\sigma(H)} \epsilon\mu(\mathrm{d}\epsilon)\right) = \int_{\sigma(H)} \mathrm{e}^{-it\epsilon}\mu(\mathrm{d}\epsilon)$$

Für die Autokorrelationsfunktion $c(t)$ gilt dann dementsprechend:

$$\begin{aligned} c(t) = \langle\psi, U(t)\psi\rangle &= \int_x \psi^*(x)U(t)\psi(x)\mathrm{d}x \\ &= \int_x \psi^*(x)\int_{\sigma(H)} \mu(\mathrm{d}\epsilon)\mathrm{e}^{-it\epsilon}\psi(x)\mathrm{d}x \\ &= \int_{\sigma(H)} \mathrm{e}^{-it\epsilon}\int_x \psi^*(x)\mu(\mathrm{d}\epsilon)\psi(x)\mathrm{d}x \\ &= \int_{\sigma(H)} \mathrm{e}^{-it\epsilon}\langle\psi, \mu(\mathrm{d}\epsilon)\psi\rangle \end{aligned} \tag{6}$$

Die inverse Fouriertransformation von $c(t)$ gibt ein moduliertes Spektrum der Energieeigenwerte von H.

$$\begin{aligned} c(\epsilon) = \mathcal{F}^*[c](\epsilon) &= \frac{1}{\sqrt{2\pi}}\int_{\mathbb{R}} c(t)\mathrm{e}^{it\epsilon}\mathrm{d}t \\ &= \frac{1}{\sqrt{2\pi}}\int_{\sigma(H)}\int_{\mathbb{R}} \mathrm{e}^{it(\epsilon-\epsilon')}\mathrm{d}t\langle\psi, \mu(\mathrm{d}\epsilon')\psi\rangle \\ &= \sqrt{2\pi}\int_{\sigma(H)} \delta(\epsilon-\epsilon')\langle\psi, \mu(\mathrm{d}\epsilon')\psi\rangle \end{aligned} \tag{7}$$

Um die Bedeutung von $c(\epsilon)$ besser zu verstehen, betrachte man einen etwas weniger allgemeinen Fall eines Systems mit isolierten Eigenwerten (H kompakt und normal da selbtstadjungiert). In einer Basis, in der dieser Hamiltonian diagonal ist, besitzt jeder Zustand eine Zerlegung $\psi = \sum_j a_j e_j$ mit Eigenfunktion e_j zu Eigenenergie ϵ_j. Die Autokorrelation ist nach (6) $c(t) = \sum_j |a_j|^2\mathrm{e}^{-it\epsilon_j}$ und das modulierte Spektrum ist damit nach (7):

$$c(\epsilon) = \sqrt{2\pi}\sum_j |a_j|^2\delta(\epsilon-\epsilon'_j) \tag{8}$$

Die δ-Distribution in der letzten Formel kommt natürlich nur dann zustande, wen über ganz $\mathbb{R}$ integriert wird. Für alle praktischen Zwecke steht aber nur ein begrenztes Intervall $I = [0:T]$ zur Verfügung. Statt des echten Spektrums bekommt man also nur die Faltung von $c(\epsilon)$ mit $\mathcal{F}[w](\epsilon)$, wobei $w(t)$ die Rechteck-Fensterfunktion mit Träger in

I ist. Nach Faltungstheorem erhält man:

$$\begin{aligned}
\mathcal{F}[cw](\epsilon) &= \int_{\mathbb{R}} c(t)w(t)\mathrm{e}^{it\epsilon}\mathrm{d}t \\
&= \int_{\mathbb{R}}\int_{\mathbb{R}} c(t)w(t')\delta(t-t')\frac{2\pi}{(\sqrt{2\pi})^2}\mathrm{e}^{it\epsilon}\mathrm{d}t\mathrm{d}t' \\
&= \frac{1}{(\sqrt{2\pi})^2}\int_{\mathbb{R}}\int_{\mathbb{R}} c(t)w(t')\int_{\sigma(H)}\mathrm{e}^{i(t-t')\epsilon'}\mathrm{d}\mu(\epsilon')\mathrm{e}^{it\epsilon}\mathrm{d}t\mathrm{d}t' \\
&= \int_{\sigma(H)}\left(\int_{\mathbb{R}}\frac{c(t)}{\sqrt{2\pi}}\mathrm{e}^{it\epsilon'}\mathrm{d}t\right)\left(\int_{\mathbb{R}}\frac{w(t')}{\sqrt{2\pi}}\mathrm{e}^{it'(\epsilon-\epsilon')}\mathrm{d}t'\right)\mathrm{d}\mu(\epsilon') \\
&= \int_{\sigma(H)}\mathcal{F}[c](\epsilon')\mathcal{F}[w](\epsilon-\epsilon')\mathrm{d}\mu(\epsilon') \\
&= (\mathcal{F}[c]*\mathcal{F}[w])(\epsilon) \qquad (9)
\end{aligned}$$

Für $w(t) = \Theta(t)\Theta(T-t)$ erhält man

$$W(\epsilon) = \mathcal{F}[w](\epsilon) = \frac{1}{\sqrt{2\pi}}\int_0^T \mathrm{e}^{it\epsilon}\mathrm{d}t = -i\frac{\mathrm{e}^{iTE}-1}{\sqrt{2\pi}E}$$

$$|W(\epsilon)| = \sqrt{\frac{1-\cos(T\epsilon)}{\pi\epsilon^2}} = \frac{|\sin(\frac{T\epsilon}{2})|}{\pi|\epsilon|}$$

Das Rechteck-Fenster verschmiert also nebeneinander liegende Peaks und verringert die Auflösung des Verfahrens. Man kann das umgehen, indem man eine andere Fenster-Funktion verwendet. Für dieses Projekt wurde die übliche von-Hann Funktion

$$w(t) = \begin{cases} \frac{1-\cos(\frac{2\pi t}{T})}{2}, & t \in I = [0:T] \\ 0, & \text{sonst} \end{cases} \qquad (10)$$

verwendet.

3 Diskrete Fouriertransformation

Für eine Simulation, die in endlicher Zeit laufen soll, kann man keine direkte Fouriertransformation benutzen, denn die symbolische Auswertung der Integrale ist leider nur für sehr wenige Probleme mit kleinen Gittern sinnvoll implementierbar. Deswegen diskretisiert man bei Simulationen normalerweise den Ort (und Zeit) und geht von kontinuierlichen zu diskreten Fouriertransformationen über.

Sei $x(t)$ ein zunächst kontinuierliches Signal. Durch Festlegen eines Samplingabstands τ erhält man den Dirac-Kamm $s(t) = \delta(t - m\tau)$. Ein diskretisiertes Signal ist dann das punktweise Produkt von $s(t)$ und $x(t)$. Unter des Annahme, dass $x(t)$ periodisch ist mit $x(t) = x(t + n\tau)$ erhält man für die für die Fouriertransformierte dann:

$$\begin{aligned}
\mathcal{F}[xs](\epsilon) &= \frac{1}{\sqrt{2\pi}}\int_{\mathbb{R}}\sum_{m=-\infty}^{\infty} x(t)\delta(t-m\tau)\mathrm{e}^{-it\epsilon}\mathrm{d}t \\
&= \frac{1}{\sqrt{2\pi}}\sum_{m=-\infty}^{\infty} x(m\tau)\mathrm{e}^{-im\tau\epsilon} \\
&= \frac{1}{\sqrt{2\pi}}\sum_{l=-\infty}^{\infty}\left(\sum_{m=0}^{n-1} x(m\tau)\mathrm{e}^{-im\tau\epsilon}\right)\mathrm{e}^{-iln\tau\epsilon} \\
&= \left(\sum_{m=0}^{n-1} x(m\tau)\mathrm{e}^{-im\tau\epsilon}\right)\cdot\frac{1}{\sqrt{2\pi}}\left(\sum_{l=-\infty}^{\infty}\mathrm{e}^{-iln\tau\epsilon}\right) \\
&= \frac{1}{N}\tilde{\mathcal{F}}[x](\epsilon)\left(\frac{\sqrt{2\pi}}{\tau}\sum_{l=-\infty}^{\infty}\delta\left(n\epsilon - l\frac{2\pi}{\tau}\right)\right)
\end{aligned}$$

Die Transformierte eines periodischen diskreten Signals ist also selbst von der gleichen Form. Insbesondere folgt aus der letzen Gleichung, dass

$$\epsilon_l = \frac{2\pi}{\tau}\frac{l}{n} = \frac{2\pi}{T}l \qquad (11)$$

und die Definition der diskreten Fourierfransformation: Sei $x \in \mathbb{C}^n$ ein Vektor dann ist die DFT von x definiert als

$$\begin{aligned}
\hat{x}_\alpha &= \tilde{\mathcal{F}}[x]_\alpha = N\sum_{m=0}^{n-1} x_m\mathrm{e}^{-im\tau\epsilon_\alpha} \\
&= N\sum_{m=0}^{n-1} x_m\mathrm{e}^{-im\tau\frac{2\pi}{n\tau}\alpha} = N\sum_{m=0}^{n-1} x_m\mathrm{e}^{-i2\pi\frac{m}{n}\alpha} \qquad (12)
\end{aligned}$$

Bemerkenswert ist, dass die Gleichung (12) in keinerlei Weise mehr von dem Samplingabstand abhängt, woraus sich die Universalität dieser Transformation ergibt. Die anfangs benutze Voraussetzung der Periodizität des Signals ist also für jede Funktion auf einem endlichen Intervall erfüllbar: man setzt die Funktion einfach periodisch fort.

Andereseits muss man bei der Simulation aufpassen, dass des Samplingbereich groß genug gewählt wird, damit die Simulation nie an den Rand des Intervalls kommen kann, was meist unerwünschte Effekte der Periodizität zur Folge hätte.

Die Inverse der DFT berechnet muss bis auf einen Normierungsfaktor die gleiche Gestalt haben:

$$x_k = \tilde{\mathcal{F}}^*[\hat{x}]_k = N\sum_{\alpha=0}^{n-1}\hat{x}_\alpha\mathrm{e}^{+i2\pi\frac{\alpha}{n}k} \qquad (13)$$

Durch Einsetzen bekommt man dann direkt:

$$\begin{aligned}\tilde{\mathcal{F}}^*[\tilde{\mathcal{F}}[x]]_k &= N^2 \sum_{\alpha=0}^{n-1}\sum_{m=0}^{n-1} x_m \mathrm{e}^{-i2\pi\frac{m}{n}\alpha}\mathrm{e}^{+i2\pi\frac{\alpha}{n}k} \\ &= N^2 \sum_{m=0}^{n-1} x_m \sum_{\alpha=0}^{n-1} \mathrm{e}^{i2\pi\frac{\alpha}{n}(k-m)} \\ &= N^2 \sum_{m=0}^{n-1} x_m n\delta_k^m = N^2 n x_k \Rightarrow N := \frac{1}{\sqrt{n}}\end{aligned}$$

4 Betrachtung einiger Potentiale

4.1 Suchalgorithmen

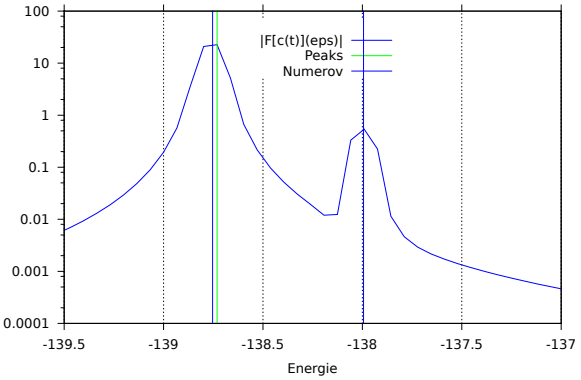
Direkte Suche

Dieser Suchalgorithmus stammt von (SAPS) und benutzt die einfachste Signifikanzfunktion S_1. Da dies meistens nicht ausreicht um alle Peaks zu finden werden nach einer Iteration alle gefundenen Peaks innerhalb des zugehörigen Fensters gelöscht (auf `nan` gesetzt), worauf eine erneute Iteration folgt, bei der allerdings die Suchschranke (Forderung in Standardabweichungen) verdoppelt wird, um eine Konvergenz des Verfahrens zu erzwingen.

Start
- berechne werte $a_i = S(d_i)$, alle Werte > 0 sind potentielle Peaks

Iteration
- vergrößere peak Fenster w um 1.5
- verdopple Suchschranke h
- berechne Mittelwert m von $\{a_i\}$
- berechne Standardabweichung s von $\{a_i\}$
- alle Werte > 0, die um das h-Fache von s abweichen im Fenster eines Peaks
- verkleinere jedes Fenster solange, bis nur der maximale Wert bleibt; dieser Wert ist der gesuchte Peak
- breche ab, falls keine Peaks gefunden wurden; andernfals lösche alle Peaks aus $\{a_i\}$ und beginne nächste Iteration

Dieser Suchalgorithmus stellte sich leider Prinzipbedingt als unzuverlässig heraus, da einerseits apriori die Standardabweichung der Peaks erraten werden muss, und andererseits, weil diese i.A. nicht über das ganze Spektrum gleich bleibt.



Wie man der oberen Abbildung entnehmen kann sind die gefundenen Peaks außerdem nicht immer nah an den tatsächlichen Energiewerten, weil durch die Diskretisierung sich das Maximum deutlich vom richtigen Wert verschieben kann.

Leider schlägt der Algorithmus manchmal sogar komplett fehl, wie beim Grundzustand des im weiteren betrachteten Kastenpotentials. Der Peak befindet sich am Rand und unterschiedet sich nicht all zu sehr von seiner Umbgebung, was dazu führt, dass der Suchalgorithmus ihn ignoriert.

Akima-Suche

Die Suche mit Hilfe der Akima Interpolation verläuft flgendermaßen: Durch die Punkte des Spektrums wird eine Akima-Interpolierende gelegt, d.h. es werden die Koeffizienten der Interpolation berechnet.

Aus der Formel für die Interpolierende

$$\begin{aligned}S_i(x) &= a_i + b_i(x-x_i) + c_i(x-x_i)^2 + d_i(x-x_i)^3 \\ S_i'(x) &= b_i + 2c_i(x-x_i) + 3d_i(x-x_i)^2 \\ S_i''(x) &= 2c_i + 6d_i(x-x_i)\end{aligned}$$

kann man direkt die möglichen Nullstellen der Ableitung berechnen:

$$x_\pm = \frac{\pm\sqrt{c_i^2 - 3b_i d_i} - c_i + 3dx_i}{3d_i} \tag{14}$$

Sofern einer dieser Werte in dem Intervall $[x_i; x_{i+1}]$ liegt, wird mit der zweiten Ableitung verifiziert, ob es sich tatsächlich um ein Maximum handelt.

Der Algorithmus ist relativ robust und findet alle Peaks ohne falsche Positive zu generieren. Allerdings leidet wegen des Verzichts auf Stetigkeit der zweiten Ableitung die Genauigkeit.

Kubische Spline-Suche

Analog zur Akima-Suche werden auch hier die Koeffizienten der Spline-Interpolierenden berechnet.

$$P_i(x) = (x - x_i)\left(\frac{a_i h_i}{6} - \frac{f_i}{h_i}\right) + \frac{(x - x_i)^3}{6h_i} a_{i+1}$$
$$+ (x - x_i)\left(\frac{f_{i+1}}{h_i} - \frac{a_{i+1} h_i}{6}\right) - \frac{(x - x_{i+1})^3}{6h_i} a_i$$
$$P_i'(x) = \left(\frac{a_i h_i}{6} - \frac{f_i}{h_i}\right) + \frac{(x - x_i)^2}{2h_i} a_{i+1}$$
$$+ \left(\frac{f_{i+1}}{h_i} - \frac{a_{i+1} h_i}{6}\right) - \frac{(x - x_{i+1})^2}{2h_i} a_i$$
$$P_i''(x) = \frac{(x - x_i)}{h_i} a_{i+1} - \frac{(x - x_{i+1})}{h_i} a_i$$

Die Nullstellen der Ableitung sind damit:

$$r = (6a_{i+1}x_i - 6a_i x_{i+1})^2 - 4(3a_i - 3a_{i+1})\cdot$$
$$\cdot\left(-a_i h_i^2 + a_{i+1} h_i^2 - 3a_{i+1} x_i^2 + 3a_i x_{i+1}^2 + 6f_i - 6f_{i+1}\right)$$
$$x_\pm = \frac{\pm\sqrt{r} - 6a_{i+1}x_i + 6a_i x_{i+1}}{6(a_i - a_{i+1})} \qquad (15)$$

So wie auch bei der Akima Interpolation wird anschließend entschieden, ob die berecheneten Punkte Maxima sind, also zu den Peaks gehören.

Obwohl dieser Algorithmus bessere Werte liefert als die Akima-Suche, Produziert er einige falsche Positive und manchaml doppelte Punkte innerhalb einzelner Peaks.

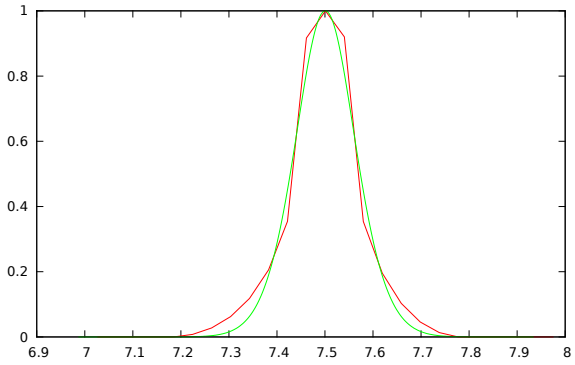
Nichtlineare Regression

Die die vorherigen Algorithmen leider keine zufriedenstellende Ergebnisse lieferten, musste ich mir völlig neues Terrain betreten und einen Weg finden, die Peaks exakter zu vermessen.

Die Idee hier ist einfach: Die Peaks im logarithmischen Spektrum liegen meistens im Verhältnis zu ihrer Breite weit außeinander. Sie können darum näherungsweise durch eine Kombination aus einem Lorentz- und Gauß-Profil beschrieben werden.

Die Modellfunktion ist also:

$$m(x, a) = \frac{a^2}{\mathrm{e}^{x^2/a^2}(a^2 + x^2)} \qquad (16)$$



Das Verfahren, das sich hier also anbietet ist, mit Hilfe der Spline-Suche ein vorläufiges Spektrum zu berechnen, und daraus einzelne Peaks auf einem vom Spektrum abhängendem Fenster zu isolieren. Anschließend Normiert man den Peak innerhalb des Fensters und minimiert die Funktion

$$\mathrm{ls}(x, a) = \sum_{i=0}^{k} (m(x_i - x, a) - f_i)^2 \qquad (17)$$

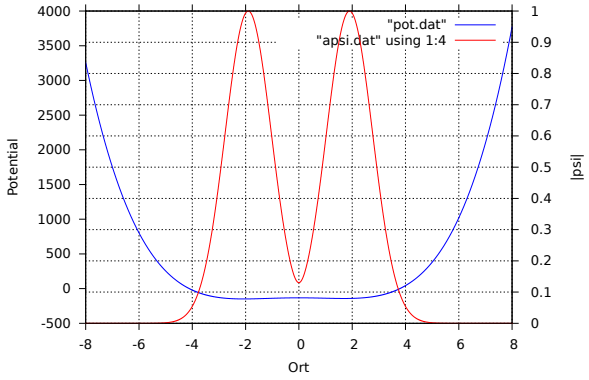
bezüglich a, wobei man für $x = x^{(0)}$ den Wert aus der Spline- oder Akima-Suche benutzt. Auf diese Weise bekommt man den Wert $a^{(0)}$, mit dem man nun ls nach x minimeiren kann und $x^{(1)}$ erhält. Falls $x^{(0)}$ nah am Peak-Mittelpunkt war, dann konvergiert dieses Verfahren seher schnell gegen den gesuchten Peak-Mittelpunkt x_m.

Die Grenzen dieses Verfahrens liegen natürlich genau dort, wo die Voraussetzung verletzt wird, dass die Peaks isoliert sind. (Deswegen dann dieses Verfahren den Grundzustand des Kastenpotentials ebenfalls nicht auflösen.)

4.2 Spektra

Wie schon in der oben angedeutet werde ich im Weiteren die gefundenen Energiewerte mit der Numerov-Approximation für die stationäre Schrödingergleichung vergleichen.

4.2.1 Asymetrischer Doppeltopf



Das erste Potential ist ein einfaches Polynom 4. Ordnung, wobei die Koeffizienten aus (FFS) entnommen wurden. Leider konnte ich die Ergebnisse von (FFS) nicht für die genannten Parameter reproduzieren, die Energien stimmen jedoch für meine Konfiguration mit den gegebenen mehr oder weniger überein. (Die Parameter aus (FFS) sind offensichtlich falsch ein $\tau = 5.7$ Entspricht einer maximalen simulierbaren Energie von $\approx \pm 0.551157$, die behaupteten Energien von ≈ -144 können also niemals aus einer Simulation mit diesen Parametern entstanden sein. Man erhält widerum ähnliche Werte wenn man sich um 4(!) Größenordnungen von den agngegebenen Parametern entfernt, d.h. $\tau = \frac{5.7}{1000}$ und $\xi = \frac{0.825}{10}$).

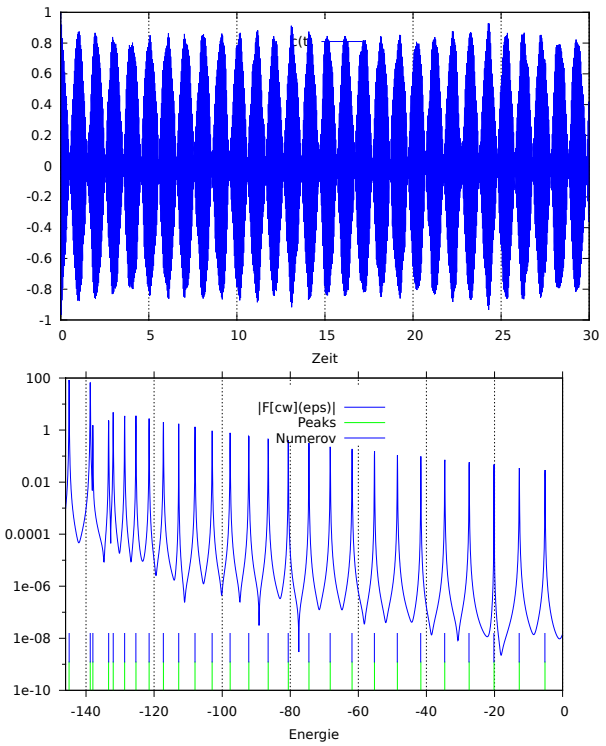
Die Parameter werden an die Simulation mit Hilfe eines einfachen Lua-Skripts übergeben, in dem die Tabelle config definiert wird.

```
local math = require("math")
local complex = require("complex")
local exp = math.exp

config = {
  bins = 4096;
  dt = 0.0001;
  range = {-8,8};
  steps = 10; runs = 150000;
  vstep = 1; vframes = 1;
  --
  potential = function(x)
    local k0, k2, k3, k4 = -132.7074997,
        7, .5, 1
    return  k0-k2*x^2+k3*x^3+k4*x^4
  end;
  psi = function(x)
    local a, s = 1.9, 0.87
    return {exp(-(x-a)^2/(2*s^2)),exp(-(x
        +a)^2/(2*s^2))}
  end;
  enrgrange = {-146, 0, 4, 1};
  output = {
    dir = "./data/simple";
    apsi = "apsi.dat";
    pot = "pot.dat";
    corr = "corr.dat";
    dftcorr = "dftcorr.dat";
    spectrum = "spectrum.dat";
    numen = "numen.dat";
    splen = "splen.dat";
    aken = "aken.dat";
    ccsen = "ccsen.dat";
  };
}
```

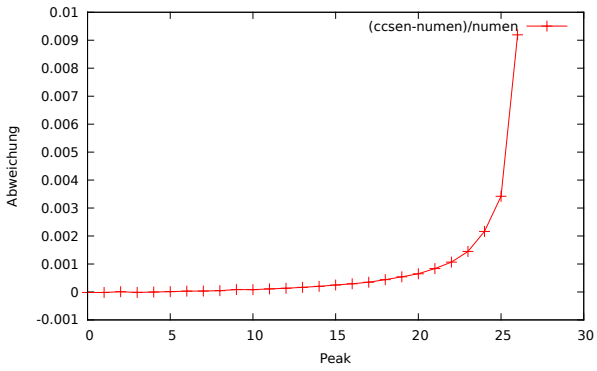
../presets/simple.lua

Die einzelnen Variablen sind (Grau = optional)

bins	Anzahl der Stützstellen im Ort
dt	einzelner Zeitschritt (τ)
range	das betrachtete Ortsintervall $[-8, 8]$
steps	Zeitschritte zwischen Korrelationsmessungen
runs	Anzahl Korrelationsmessungen
vstep	Zeitschritte zwischen Frames
vframes	Anzahl Frames
potential(x)	reelles Potential ($V(x)$)
psi(x)={u,v}	Wellefunktion $\psi(x) = u + iv$
energie(k)	exakte Energien ϵ_k
enrgrange	betrachteten Energien: Min.,Max., Peak-Fenster(in Punkten), Suchschranke
output	
.dir	Ausgabeverzeichnis
.apsi	Werte von $\psi(x, t = 0)$
.pot	Werte von $V(x)$
.corr	Werte von $c(t)$
.dftcorr	Werte von $\mathcal{F}[cw](\epsilon)$
.spectrum	Peaks: direkte Suche
.numen	Numerov Approximation
.splen	Peaks: kubische Spline-Suche
.aken	Peaks: Akima-Suche
.ccsen	Peaks: nichtlineare Regression

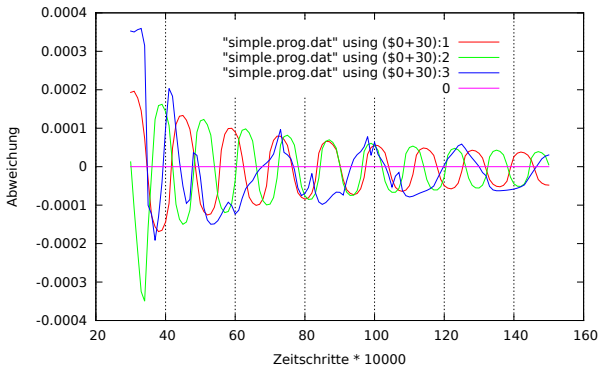


Peaksuche	Numerov
-144.9689052	-144.9662986
-138.7558697	-138.7531872
-137.9930795	-137.9943586
-133.3565262	-133.3544215
-132.0175649	-132.017109
-128.6536469	-128.6551459
-125.3361586	-125.3398237
-121.4695901	-121.4735992
-117.2722548	-117.2776404
-112.7618149	-112.7713057
-107.9795671	-107.9886065
-102.9412753	-102.9525932
-97.66926161	-97.68213261
-92.17758948	-92.19280618
-86.48045195	-86.49781338
-80.58861207	-80.60852286
-74.51311105	-74.53485882
-68.26174335	-68.28557958
-61.84119739	-61.86848533
-55.26062198	-55.29057713
-48.5266798	-48.55818105
-41.64223027	-41.67704696
-34.61546639	-34.6524282
-27.44933412	-27.48914674
-20.14794992	-20.19164709
-12.72036832	-12.76404124
-5.162248281	-5.210146622



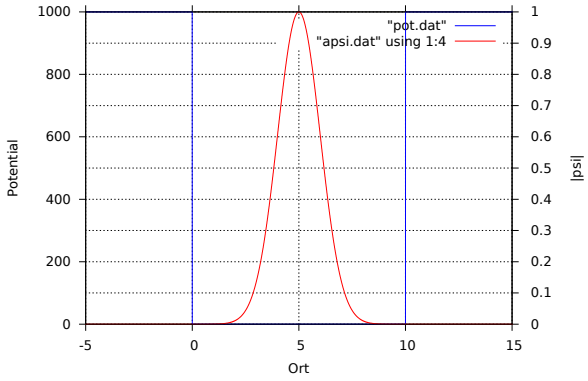
Wie man aus dem letzen Graphen entnehmen kann, wächst der relative Fehler für größere Energien an.

Konvergeznverhalten



Die Abweichung vom Numerov Wert wird mit der Anzahl der Zeitschritte kleiner, oszilliert aber mit einer vom Peak abhängender Frequenz.

4.2.2 Potentialkasten



Als nächstes versuche ich bekannte Erbgebnisse für einen unendlich tiefen Potentialkasten näherungsweise mit einem endlich hohen Potentialkasten zu approximieren.

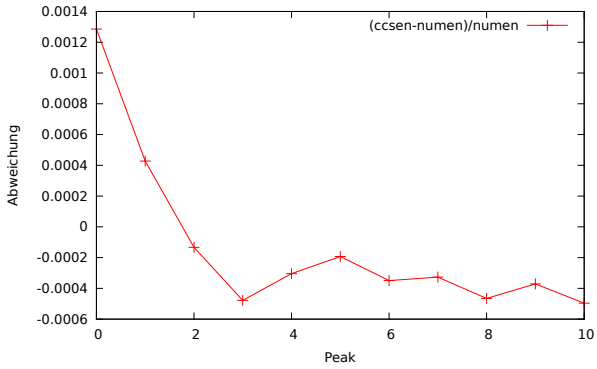
Die konfiguration für die Simulation lautet dementsprechend:

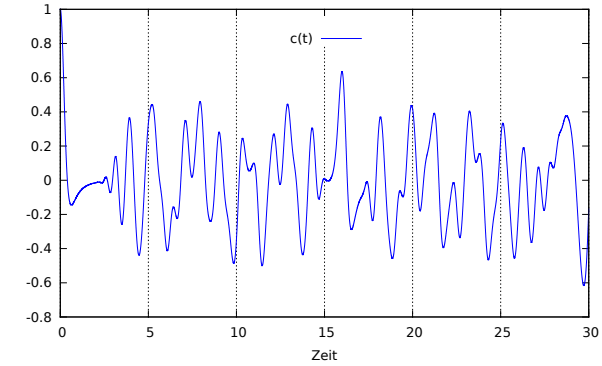
```
local math = require("math")
local complex = require("complex")
local L = 10
local exp = math.exp
local cexp = complex.exp
local pi = math.pi

config = {
  bins = 4096;  dt = 0.0001;
  range = {-5,15};
  steps = 10; runs = 150000;
  vstep = 100; vframes = 200;
  --
  potential = function(x)
    if 0 < x and x < L then return 0 else
        return 1000 end
  end;
  psi = function(x)
    return exp(-(x-L/2)^2/(2)) * cexp({0,
        2 * x})
  end;
  energy = function(r)
    local k = r + 1
```

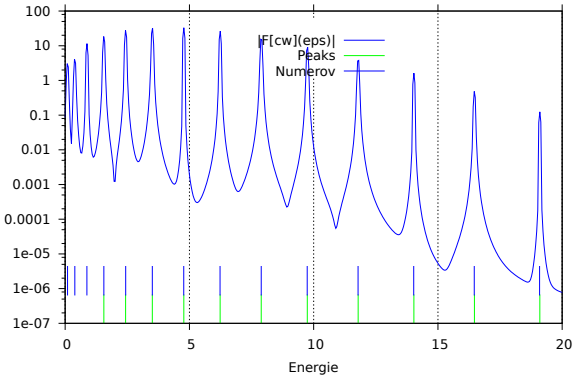
```
    return pi^2 * k^2/L^2
  end;
  enrgrange = {0, 20, 4, 2};
  output = {
    dir = "./data/square";
    apsi = "apsi.dat";
    pot = "pot.dat";
    corr = "corr.dat";
    dftcorr = "dftcorr.dat";
    spectrum = "spectrum.dat";
    numen = "numen.dat";
    splen = "splen.dat";
    aken = "aken.dat";
    ccsen = "ccsen.dat";
  };
}
```

../presets/square.lua

Peaksuche	Numerov	exakt
	0.0974096378	0.09869604401
	0.3896383141	0.394784176
	0.8766853172	0.8882643961
1.556546034	1.55854946	1.579136704
2.434187902	2.435229079	2.4674011
3.507189004	3.506722029	3.553057584
4.775307245	4.773025686	4.836106157
6.236034546	6.23413694	6.316546817
7.891574923	7.890052188	7.994379565
9.744172522	9.740767336	9.869604401
11.79013325	11.78627779	11.94222133
14.033112	14.02657845	14.21223034
16.46776653	16.46166369	16.67963144
19.10099893	19.09152739	19.34442463

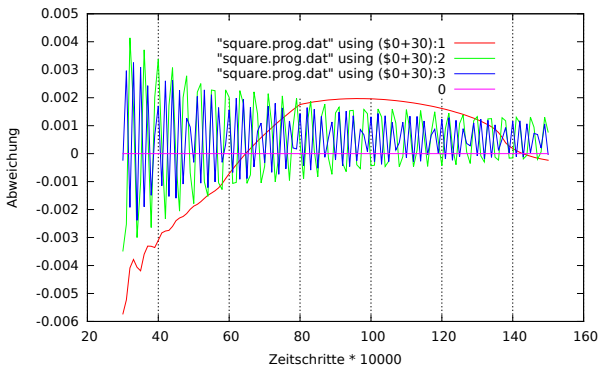




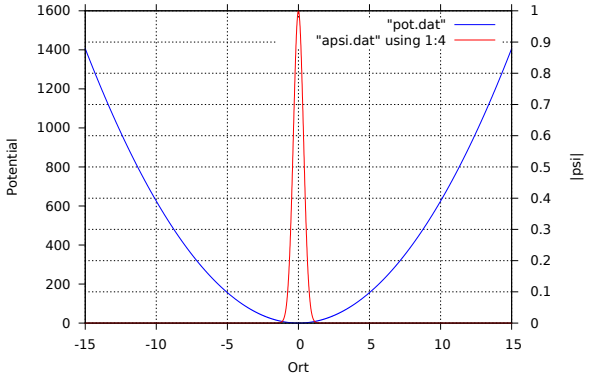


Der relative Fehler wird im Gegensatz zum vorgerigen Potential kleiner für größere Energien. Wie schon eingangs erwähnt, kann mein Verfahren die ersten drei Energien nicht auflösen. Die Ergebnisse der reinen Spline-Suche bzw. Akima-Suche geben zwar einen Wert, dieser schwankt aber signifikant um den erwarteten.

Konvergeznverhalten



Harmonischer Oszillator

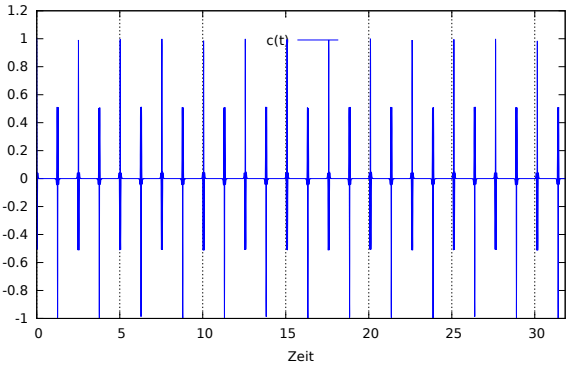


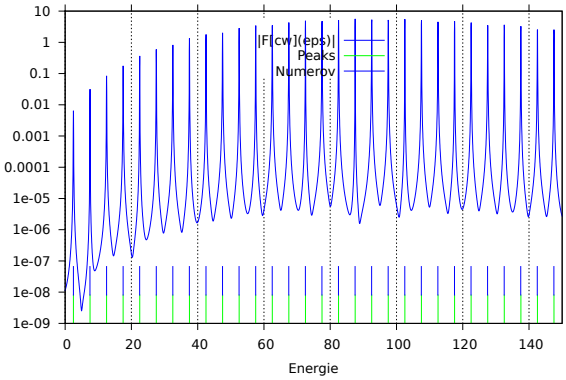
Als letrztes Potential betrachte ich den Harmonischen Oszillator, da für ihn die Exakte werte bekannt sind. In der Konfugurationsdatei werden diese mit der Funktion `energy` festegelgt:

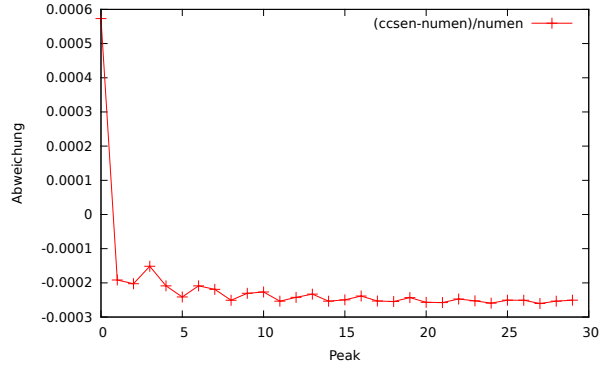
```
local math = require("math")
local complex = require("complex")
local omega = 5;
local exp = math.exp
local sqrt = math.sqrt
local pi = math.pi

config = {
  bins = 4096; dt = 1 / (1000 * pi);
  range = {-15,15};
  steps = 10; runs = 50000;
  vstep = 100; vframes = 200;
  --
  potential = function(x)
    return omega^2/4 * x ^ 2;
  end;
  psi = function(x)
    local a = .5
    local aa = a * a
    local x0 = 0
    local k0 = 10
    local xx = (x-x0) * (x-x0)
    return  math.exp(-xx/(aa)) * complex.
        exp({0, k0*(x)})
  end;
  energy = function(k)
    return omega * (.5 + k)
  end;
  enrgrange = {0, 150, 6, 1};
  output = {
    dir = "./data/harmosca";
    apsi = "apsi.dat";
    pot = "pot.dat";
    corr = "corr.dat";
    dftcorr = "dftcorr.dat";
    spectrum = "spectrum.dat";
    numen = "numen.dat";
    splen = "splen.dat";
    aken = "aken.dat";
    ccsen = "ccsen.dat";
  };
}
```

../presets/harmosca.lua

Peaksuche	Numerov	exakt
2.498567361	2.5	2.5
7.50143752	7.499999998	7.5
12.50253132	12.49999999	12.5
17.50264914	17.49999998	17.5
22.50469515	22.49999995	22.5
27.50663461	27.49999992	27.5
32.50678341	32.49999987	32.5
37.50820743	37.4999998	37.5
42.51065952	42.49999971	42.5
47.51095092	47.49999959	47.5
52.51189857	52.49999945	52.5
57.51458301	57.49999928	57.5
62.5151468	62.49999908	62.5
67.51572909	67.49999884	67.5
72.51838651	72.49999856	72.5
77.51935027	77.49999824	77.5
82.51967097	82.49999788	82.5
87.52213463	87.49999748	87.5
92.52353604	92.49999702	92.5
97.52370097	97.49999651	97.5
102.5263325	102.4999959	102.5
107.5276827	107.4999953	107.5
112.5277982	112.4999946	112.5
117.5296679	117.4999939	117.5
122.531768	122.4999931	122.5
127.5319428	127.4999922	127.5
132.5332312	132.4999913	132.5
137.535769	137.4999902	137.5
142.5361158	142.4999891	142.5
147.5369614	147.4999879	147.5

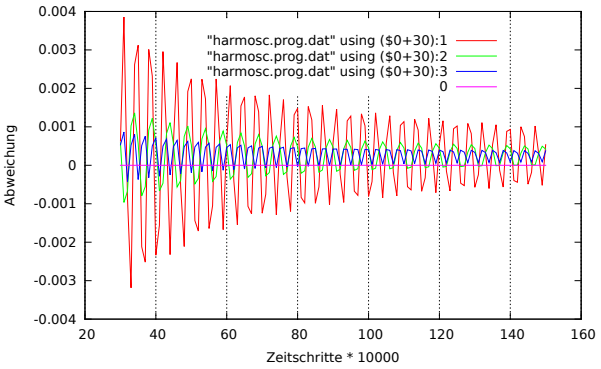






Auch bei diesem Potential verringert sich der relative Fehler für größere Energiewerte.

Konvergeznverhalten



Die Abweichung vom Numerov Wert wird auch für dieses Potential kleiner und auch hier ist eine deutliche Überlagerung mit einer Schwingung zu sehen.

5 Fazit

Wie man aus der betrachtung der Spektra entnehmen kann, erlaubt die Autokorrelations-Methode immer die grobe Bestimmung eines Spektrums. Auch behält dieses Spektrum bei kleinen Veränderungen der Simulationsparamter seine Form.

Für genaue Approximation von des Spektrums eignet sich diese Verfahren leider nicht: Das Konvergenzverhalten ist so, dass jeder Peaks mit einer eigenen Frequenz oszilliert, die man bei der Suche nicht berücksichtigen kann, da man sie apriori nicht wissen kann.

6 Quellkode

Der Quellkode ist unter `https://github.com/xaberus/project3` veröffentlicht und lässt sich mit `git clone git://github.com/xaberus/project3.git` herunterladen.

Quellen

[FFS] M.D Feit, J.A Fleck Jr., A Steiger *Solution of the Schrödinger equation by a spectral method* `http://dx.doi.org/10.1016/0021-9991(82)90091-2`

[FMF] J. A. Fleck, J. R. Morris and M. D. Feit *Time-dependent propagation of high energy laser beams through the atmosphere* `http://dx.doi.org/10.1007/BF00896333`

[MA] Mrinal Mandal, Amir Asif *Continuous and Discrete Time Signals and Systems* Kap.12.1 `http://www.cambridge.org/resources/0521854555/4421_Chapter%2012%20-%20Discrete%20Fourier%20transform.pdf`

[SAPS] Girish Keshav Palshikar *Simple Algorithms for Peak Detection in Time-Series* `http://www.tcs-trddc.com/trddc_website/pdf/SRL/Palshikar_SAPDTS_2009.pdf`