Commun. Math. Phys. 106, 495–532 (1986)

Communications in Mathematical Physics

A Non-Gaussian Renormalization Group Fixed Point for Hierarchical Scalar Lattice Field Theories

Hans Koch[1] and Peter Wittwer[2,3]

Department of Mathematics, Rutgers University, New Brunswick, NJ 08903, USA

Abstract. A rigorous method is developed to handle the "large field problems" in the Wilson-Kadanoff renormalization group approach to critical lattice systems of unbounded spins. We use this method to study in a hierarchical approximation the non-Gaussian renormalization group fixed point which governs the infrared behaviour of critical lattice field theories in three dimensions. The method is an improvement of the analyticity techniques of Gawedzki and Kupiainen: using Borel summation techniques we are able to incorporate the "large field region" into the "perturbative region" so that the theory is completely described in terms of convergent expansions.

1. Introduction

A major obstacle in the analysis of critical statistical mechanics systems is the lack of small expansion parameters. For example very little is known about scalar lattice spin systems with non-Gaussian critical long distance behavior

$$\int d\mu(\phi)\phi_i\phi_j \sim \frac{1}{|i-j|^{d-2+\eta}}, \quad |i-j|\to\infty, \quad \eta \geqq 0. \tag{1.1}$$

Here, $d\mu$ is a translation invariant Gibbs measure on configurations $\phi: \mathbb{Z}^d \to \mathbb{R}$. Although renormalization group (RG) concepts provide a powerful framework to think about such problems, it has been impossible so far to convert them into rigorous results, except for certain limit cases [10–12].

The methods presented in this paper are not restricted to such cases. However, we restrict our analysis to a class of hierarchical models. Such models have a long history in the testing of RG ideas [13–20, 25, 26]. We prove here the existence of a

[1] Supported in part by the National Science Foundation under Grant No. DMS-8540879
[2] Supported in part by the National Science Foundation under Grant No. DMR81-14726
[3] Part of this work has been carried out during a visit of the author at the Courant Institute of Mathematical Sciences, NYU

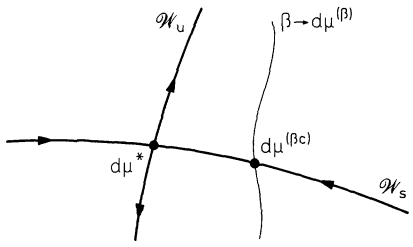

Fig. 1. The fixed point $d\mu^*$ for $\mathcal{N}$ and its stable and unstable manifold

non-Gaussian hierarchical fixed point for a Wilson-Kadanoff RG transformation, in $d=3$ dimensions. This type of transformation acts on a measure in two steps. First, some short range degrees of freedom (those which are thought to be irrelevant) are integrated out. Then the system is rescaled in a way that is consistent with a long distance behavior as described in Eq. (1.1). In a suitable space of measures, the RG transformation $\mathcal{N}$ is believed to behave as shown in Fig. 1.

Here, $d\mu^*$ is a non-Gaussian hyperbolic fixed point for $\mathcal{N}$, with an unstable manifold $\mathcal{W}_u$ of dimension one. The codimension one stable manifold $\mathcal{W}_s$ through $d\mu^*$ represents the set of critical measures. These measures approach $d\mu^*$ under iteration of $\mathcal{N}$, and from this setup it follows that they all have the same long distance behavior. If such a picture is correct, a critical point β_c will be observed in smooth one parameter families $\beta \mapsto d\mu^\beta$ passing near $d\mu^*$. Furthermore, if the linearization $D\mathcal{N}$ of $\mathcal{N}$ at $d\mu^*$ is analyzed, it is possible to make precise predictions about the rate of convergence or divergence (critical indices) of certain physical quantities as β approaches β_c. These rates are universal (i.e. independent of the particular one parameter family considered), since every family intersecting $\mathcal{W}_s$ is mapped arbitrarily close to $\mathcal{W}_u$ under iteration of $\mathcal{N}$, so that the behavior near β_c is controlled by the local properties of $\mathcal{N}$ near $d\mu^*$. We do not attempt here to prove the correctness of Fig. 1, except for the existence of the fixed point $d\mu^*$.

Consider now probability measures of the form

$$d\mu(\phi) = d\mu_C(\phi) F(\phi), \tag{1.2}$$

where $d\mu_C$ denotes the Gaussian measure on the space $\mathcal{D}^*$ of configurations $\phi : \mathbb{Z}^d \to \mathbb{R}$, with covariance C. Here, C is some fixed positive operator on $l_2(\mathbb{Z}^d)$, as for example the inverse lattice Laplacean.

We specify a RG transformation by choosing an "averaging operator" B and a "scaling parameter" $\alpha > 0$ such that

$$\Gamma = C - \alpha^2 B^* C B \tag{1.3}$$

is positive. Given such a choice, we define the RG operator $\mathcal{N}$ by the equation

$$\mathcal{N}(d\mu) = d\tilde{\mu} \equiv d\mu_C(\cdot)\tilde{F}(\cdot), \tag{1.4}$$

where

$$\tilde{F}(\phi) = \int d\mu_\Gamma(\psi) F(\alpha B^* \phi + \psi). \tag{1.5}$$

This definition realizes, for suitable choices of B and α, the above mentioned desiderata for a RG transformation. It is motivated by the identity

$$\int d\mu_C(\phi) F(\phi) = \int d\mu_C(\phi) \left(\int d\mu_\Gamma(\psi) F(\alpha B^* \phi + \psi) \right), \tag{1.6}$$

which shows that the image measure $d\tilde{\mu}$, which we get by integrating out fluctuations (integration with respect to $d\mu_\Gamma$) and by rescaling, is again a probability measure.

The following choice of B is tailored to the hierarchical model which is defined below. We fix L to be an integer larger than one, and we set

$$B = L^{d/2} A^* A^2 , \tag{1.7}$$

for

$$(A\phi)_i = L^{-d/2} \sum_{j:[L^{-1}j]=i} \phi_j . \tag{1.8}$$

Here, $[x]$ is the point in $\mathbb{Z}^d$ obtained from $x \in \mathbb{R}^d$ by taking the integer part of its coordinates. Note that A^*A is the projection in $l_2(\mathbb{Z}^d)$ onto functions which are constant on blocks of size L^d, and that $AA^* = \mathrm{Id}$.

Fix now

$$l = L^d , \quad \alpha = L^{-(d-2)/2} . \tag{1.9}$$

Then our class of hierarchical models is defined by choosing the covariance C to be

$$C_{ij} = \frac{l}{2}\left(\delta_{ij} + L^{-2} \sum_{n=2}^{\infty} \alpha^{2n} \delta_{[L^{-n}i],[L^{-n}j]} \right). \tag{1.10}$$

This covariance does not describe a very "realistic" spin-spin-interaction, since it is not translation invariant. However, it mimicks the long distance behavior of $(-\Delta)^{-1}$. For the fluctuation covaraince (1.3) we obtain from Eqs. (1.7) and (1.8)

$$\Gamma = \frac{l}{2} \cdot \mathrm{Id} . \tag{1.11}$$

In order to get a first impression of this particular RG transformation, let us apply $\mathcal{N}$ to a measure

$$d\mu(\phi) = d\mu_C(\phi) \exp(-\tfrac{1}{2}\langle \phi, M\phi \rangle - \mathrm{const}), \tag{1.12}$$

for M a nonnegative bounded operator on $l_2(\mathbb{Z}^d)$. A short exercise in Gaussian integrals shows that $d\tilde{\mu}$ is again of the form (1.12), but with M replaced by

$$\tilde{M} = \alpha^2 BM(1 + \Gamma M)^{-1} B^* . \tag{1.13}$$

By using this relation, the domain of $\mathcal{N}$ can be extended to include measures which are not necessarily continuous with respect to $d\mu_C$. Consider e.g. the family of Gaussians, formally corresponding to

$$M = \lambda \cdot 2A^* A , \quad \lambda \geqq 0 . \tag{1.14}$$

Inserting (1.11) and (1.14) into (1.13), we get

$$\tilde{M} = \tilde{\lambda} \cdot 2A^* A , \quad \tilde{\lambda} = \frac{L^2 \lambda}{\lambda l + 1} . \tag{1.15}$$

The action of the map induced by $\mathcal{N}$ in λ-space is shown in Fig. 2.

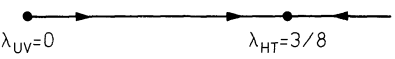

Fig. 2. The "ultraviolet" and "high-temperature" fixed points for the map $\lambda \mapsto \tilde{\lambda}$ in the case $L=2$

It is an important feature of this RG transformation that the high temperature fixed point has a finite mass ($\lambda_{\mathrm{HT}} < \infty$ in Fig. 2). It reflects the fact that the "irrelevant" degrees of freedom integrated out in (1.5) are not just the fluctuations of wavelength $< L$.

In the search for a nontrivial fixed point we will now focus on a class of measures of the form

$$d\mu(\phi) = d\mu_G(\phi) \prod_j g_B(((A^* A\phi)_j)^2), \tag{1.16}$$

where

$$G^{-1} = C^{-1} + M, \tag{1.17}$$

with M given by (1.14) for some fixed $\lambda > 0$. To simplify the notation we have omitted at this point any reference to an infrared cutoff. The functions g_B in (1.16) are chosen to be entire analytic, with a growth rate restricted in such a way that the mass term $\langle \phi, M\phi)$ dominates for large fields ϕ. Using (1.4), ..., (1.11) we obtain

$$d\tilde{\mu}(\phi) = d\mu_G(\phi) \prod_j \tilde{g}_B(((A^* A\phi)_j)^2), \tag{1.18}$$

where

$$\tilde{g}_B(t^2) = \exp\left(\left(\lambda - \frac{L^2\lambda}{\lambda l + 1}\right) t^2\right) \frac{1}{Z} \int_{-\infty}^{\infty} ds \exp(-(1+\lambda l)s^2) \left[g_B\left(\left(\frac{\alpha}{1+\lambda l} t + s\right)^2\right)\right]^l, \tag{1.19}$$

for some normalization constant Z. Our main goal is to prove that the reduced RG transformation $g_B \mapsto \tilde{g}_B$ has a non-Gaussian fixed point in $d=3$ dimensions, for $L=2$ and $\lambda = 1/3$.

The remaining part of the paper is organized as follows. In Sect. 2 we present our main results and a theorem on Borel transformations, which we need in later sections. Section 3 contains an outline of the proofs. In Sect. 4 we construct the infinite volume limit of our model and analyze the correlation functions. Section 5, finally, contains the details of our proofs in form of a computer program (written in the programming language *C* [23]).

2. Results

Motivated by the reasons mentioned in the introduction, we consider here the fixed point equation $\mathcal{T}_B h_B = h_B$, for a renormalization transformation of the form

$$\begin{aligned}(\mathcal{T}_B h_B)(t^2) = \tilde{h}_B(t^2) \equiv \exp\left(\left(\lambda - \frac{L^2\lambda}{\lambda l + 1}\right) t^2\right) \\ \times \frac{1}{c} \int_{-\infty}^{\infty} ds \exp(-(1+\lambda l)s^2) \left[h_B\left(\left(\frac{\alpha}{1+\lambda l} t + s\right)^2\right)\right]^l.\end{aligned} \tag{2.1}$$

The independent parameters d (dimension), L (linear block size) and λ (mass counterterm) will be fixed as follows

$$d=3, \quad L=2, \quad \lambda=\tfrac{1}{3}. \tag{2.2}$$

Then the block volume l and the scaling parameter α take the values

$$l=L^d=8, \quad \alpha=L^{-(d-2)/2}=\sqrt{2}/2. \tag{2.3}$$

For convenience later on we choose the normalization constant c in (2.1) to be

$$c=c(h_B)=\int_{-\infty}^{\infty} ds \exp(-(1+\lambda l)s^2)\,[h_B(s^2)]^l, \tag{2.4}$$

so that $\tilde{h}_B(0)=1$. Note that the fixed points of (2.1) and (1.19) agree up to a constant factor.

In order to complete the definition of $\mathcal{T}_B$, we will now specify its domain.

Definition 2.1. For each $\varrho>0$, we define $\mathcal{E}_\varrho$ to be the Banach space of all functions h_B which are entire analytic, real valued on real points, and for which the norm

$$\|h_B\|_\varrho=\sup_t |h_B(t)| \exp\left(-\frac{|t|}{\varrho}\right) \tag{2.5}$$

is finite. Furthermore, we define $\mathcal{E}_\varrho^+=\mathcal{E}_\varrho\backslash\{0\}$.

Proposition 2.2. *Let ϱ be a real constant, satisfying $10/3<\varrho<22$. Then there exists some positive constant $\sigma<\varrho$ such that $\mathcal{T}_B$ is a C^∞ map from $\mathcal{E}_\sigma^+$ to $\mathcal{E}_\varrho^+$.*

Proof. Let ϱ be fixed according to the above conditions. Then for every pair (σ, h_B) satisfying

$$\tfrac{10}{3}<\sigma<\varrho, \quad h_B\in\mathcal{E}_\sigma^+, \tag{2.6}$$

the integrals (2.1) and (2.4) converge and $c(h_B)>0$. We shall now determine those choices (2.6) for which $\tilde{h}_B\in\mathcal{E}_\varrho$. By (2.2) and (2.3) we have

$$\tilde{h}_B(t^2)=\frac{1}{c(h_B)}\exp(-\tfrac{1}{33}t^2)\int_{-\infty}^{\infty} ds \exp(-\tfrac{11}{3}s^2)\left[h_B\left(\left(\frac{3\sqrt{2}}{22}t+s\right)^2\right)\right]^8. \tag{2.7}$$

Using the fact that

$$\begin{aligned} |h_B((r+s)^2)| &\leqq \|h_B\|_\sigma \cdot \exp\left(\frac{|r+s|^2}{\sigma}\right) \\ &\leqq \|h_B\|_\sigma \cdot \left(\exp\left(\frac{(|r|+s)^2}{\sigma}\right)+\exp\left(\frac{(|r|-s)^2}{\sigma}\right)\right), \end{aligned} \tag{2.8}$$

for $r\in\mathbb{C}$ and $s\in\mathbb{R}$, a short computation leads to the inequality

$$|\tilde{h}_B(t^2)| \exp\left(-\frac{|t^2|}{\varrho}\right) \leqq \text{const} \cdot \exp\left(|t|^2\left(\frac{1}{\tilde{\sigma}}-\frac{1}{\varrho}\right)\right), \tag{2.9}$$

with $\tilde{\sigma}$ given by

$$\frac{1}{\tilde{\sigma}}=\tfrac{1}{33}+\tfrac{9}{242}\left(\frac{8}{\varrho}+\frac{64}{\varrho^2}\left(\tfrac{11}{3}-\frac{8}{\varrho}\right)^{-1}\right). \tag{2.10}$$

It is now easy to check that $\tilde{\sigma} > \varrho$ for σ sufficiently close to ϱ. This implies that $\mathscr{T}_B \mathscr{E}_\sigma^+ \subset \mathscr{E}_\varrho^+$. The same estimates can be used to show that $\mathscr{T}_B$ is C^∞. □

Note that $\mathscr{E}_\varrho$ is a subspace of $\mathscr{E}_\sigma$ for $\sigma < \varrho$. In fact the injection $\mathscr{E}_\varrho \mapsto \mathscr{E}_\sigma$ is compact (see e.g. [22]). As a consequence we have the following result.

Corollary 2.3. *Assume ϱ satisfies $10/3 < \varrho < 22$, and let h_B be an element of $\mathscr{E}_\varrho^+$. Then the tangent map $D\mathscr{T}_B$ of $\mathscr{T}_B$ at h_B is a compact linear operator on $\mathscr{E}_\varrho$.*

For more information on $D\mathscr{T}_B$ see below. We shall now state the main result of this paper.

Theorem 2.4. *For $\varrho = 7/2$ there exists a non-Gaussian fixed point h_B^* of $\mathscr{T}_B$ in $\mathscr{E}_\varrho^+$.*

Remark. With the above choice of constants $\lambda = 1/3 > 2/7 = 1/\varrho$, the function

$$f^*(t) = \text{const} \cdot e^{-\lambda t} h_B^*(t) \tag{2.11}$$

is exponentially decreasing along the positive real axis, i.e. the mass term dominates the large field behavior of the (finite volume) measure (1.16). In Sect. 4 we will use this fact to control the thermodynamic limit.

We prove Theorem 2.4 by first designing an algorithm which reduces the assertion to a large but finite set of inequalities (see Sect. 3). Then we use a computer[1] to generate and verify the inequalities. The listing of the computer program is given in Sect. 5. As a by-product of this method we get very good bounds on the function h_B^*. We also have the following *nonrigorous* numerical results (which we believe can be proved by the same methods used here and in [1, 3]).

Numerical Results 2.5. For $\varrho = 7/2$, the spectrum of the tangent map $D\mathscr{T}_B$ of $\mathscr{T}_B$ at the fixed point h_B^* is contained inside the unit disk, with the exception of a simple eigenvalue $\delta = 2.9040714988\ldots$.

From these data we get, using the relations given in [18, 19], the following critical indices

$$\begin{aligned} \gamma &= \frac{2 \log L}{\log \delta} = 1.300\ldots \quad (\gamma_I \approx 1.250\ldots), \\ \nu &= \frac{\log L}{\log \delta} = 0.650\ldots \quad (\nu_I \approx 0.638\ldots), \\ \eta &= 0 \qquad\qquad\qquad\quad (\eta_I \approx 0.041\ldots). \end{aligned} \tag{2.12}$$

For a comparison we have listed in parenthesis the alleged values of the critical indices for the Ising model, calculated numerically by various authors [27–36, 38–47].

The main idea behind the proof of Theorem 2.4 is to consider h_B and $\tilde{h}_B$ to be the *Borel transform* [6, 9, 10] of two functions h and $\tilde{h}$, and then to study the fixed point problem for the RG transformation

$$\mathscr{T} : h \mapsto \tilde{h}. \tag{2.13}$$

[1] An IBM Personal Computer equipped with the 8087-math-coprocessor, and the Microsoft *C* compiler 3.0 running on top of MS-DOS 3.1

Note that this is exactly the opposite of what is normally done in perturbation theory, which is usually carried out around the trivial Gaussian fixed point of $\mathscr{T}_B$ [8, 9]. There the Borel transforms of h_B and $\widetilde{h_B}$ are studied, and not the *inverse Borel transform* as we do. The main advantage of our method is that it eliminates the need to distinguish between large fields and small fields, which has been the principal source of technical difficulties in [11, 12]. In fact the fields in the inverse Borel plane are bounded, and contrary to what one might expect, the new transformation $\mathscr{T}$ takes a very simple form.

Since our definition of the Borel transformation differs slightly from the standard one, we now first discuss some domain questions before we explicitly define the action of $\mathscr{T}$. The domain of $\mathscr{T}$ will be chosen among the following function spaces.

Definition 2.6. Given a positive number ϱ, define $\mathscr{A}_\varrho$ to be the Banach space of all functions h, which are analytic on $D_\varrho = \{z \in \mathbb{C} \,|\, |z| < \varrho\}$, continuous on the boundary ∂D_ϱ, real valued on real points, equipped with the norm

$$\|h\|_\varrho = \sup_{x \in D_\varrho} |h(x)| \,. \tag{2.14}$$

Furthermore, we define $\mathscr{A}_\varrho^+ = \mathscr{A} \backslash \{0\}$.

Note that the functions in $\mathscr{A}_\varrho$, as well as those in $\mathscr{E}_\varrho$, are uniquely determined by their Taylor series about the origin. Before defining the Borel transform, let us list first some simple bounds on Taylor coefficients. Consider two functions $g \in \mathscr{A}_\sigma$ and $h_B \in \mathscr{E}_\sigma$ for some $\sigma > 0$, and suppose that

$$g(x) = \sum_{n=0}^{\infty} c_n x^n \,, \qquad |x| < \sigma \,, \tag{2.15}$$

$$h_B(t) = \sum_{n=0}^{\infty} b_n t^n \,, \qquad t \in \mathbb{C} \,. \tag{2.16}$$

Then for all positive $\tau < \sigma$ and for all $k \in \mathbb{N}$ we have

$$|c_k| \leqq \sigma^{-k} \|g\|_\sigma \,, \qquad \|g\|_\tau \leqq \sum_{n=0}^{\infty} |c_n| \tau^n \,, \tag{2.17}$$

$$|b_k| \leqq \frac{\kappa \sqrt{k}}{\sigma^k k!} \|h_B\|_\sigma \,, \qquad \|h_B\|_\tau \leqq \kappa \sum_{n=0}^{\infty} |b_n| \frac{\tau^n n!}{\sqrt{n}} \,, \tag{2.18}$$

where κ is some constant, independent of g and h_B. The first two estimates are trivial. The third bound follows by using Cauchy's formula with contour $|t| = n\sigma$, and then Stirling's formula. The last inequality is obtained computing the norm of monomials $t \mapsto t^n$.

We define now our Borel transform $\mathscr{B}$, which differs slightly from the standard one. (see e.g. [6] for the standard definition). To make the difference more transparent, we define $\mathscr{B}$ in terms of its action on power series. An integral representation will be given in Proposition 2.8.

Definition 2.7.[2] For fixed $\sigma>0$ consider the spaces

$$e_\sigma=\bigcap_{\varrho<\sigma}\mathscr{E}_\varrho\,,\qquad a_\sigma=\bigcap_{\varrho<\sigma}\mathscr{A}_\varrho\,. \tag{2.19}$$

Assume that $h\in a_\sigma$ has the Taylor series

$$h(x)=\sum_{n=0}^{\infty}c_n x^n\,. \tag{2.20}$$

Then we define its Borel transform $\mathscr{B}h\in e_\sigma$ by

$$(\mathscr{B}h)(t)=\sum_{n=0}^{\infty}c_n\frac{(-\frac{1}{2})!}{(n-\frac{1}{2})!}t^n\,,\qquad t\in\mathbb{C}\,. \tag{2.21}$$

Note that from (2.17), (2.18) it follows that $\mathscr{B}$ is a one-to-one map from a_σ to e_σ. If a function $h_B\in e_\sigma$ is written in the form (2.16), then its inverse Borel transform is given by (2.15), with

$$c_n=\frac{(n-\frac{1}{2})!}{(-\frac{1}{2})!}b_n\,. \tag{2.22}$$

Proposition 2.8. *Assume that $0<\varrho<\sigma$. Then there is a constant κ such that for $g\in\mathscr{A}_\varrho$ and $h_B\in\mathscr{E}_\sigma$,*

$$(\mathscr{B}g)(t)=g(0)+t\int_0^\infty\frac{dx}{x^{1/2}(1+x)^{3/2}}\times\left\{\frac{1}{2\pi i}\oint_{|z|=\varrho}\frac{dz}{z}\exp\left(\frac{t}{(1+x)z}\right)\frac{g(z)-g(0)}{z}\right\}, \tag{2.23}$$

$$\|\mathscr{B}g\|_\varrho\leqq\kappa\|g\|_\varrho\,, \tag{2.24}$$

$$(\mathscr{B}^{-1}h_B)(x)=\frac{1}{\sqrt{\pi}}\int_{-\infty}^{\infty}dt\exp(-t^2)h_B(xt^2)\,, \tag{2.25}$$

$$\|\mathscr{B}^{-1}h_B\|_\varrho\leqq\kappa\|h_B\|_\sigma\,. \tag{2.26}$$

Proof. Assume that the above conditions are satisfied. Then it follows from (2.17) and (2.18) that for $\varrho<\tau<\sigma$

$$\|\mathscr{B}^{-1}h_B\|_\varrho\leqq\text{const}\,\|h_B\|_\tau\leqq\text{const}\,\|h_B\|_\sigma\,. \tag{2.27}$$

This proves (2.26). To show (2.25) we use the fact that in $\mathscr{E}_\tau$ the function h_B can be approximated by polynomials. Thus it is sufficient to consider $h_B(t)=t^n$. Using the identity $\sqrt{\pi}=(-1/2)!$, we obtain

$$\frac{1}{\sqrt{\pi}}\int_{-\infty}^{\infty}dt\,e^{-t^2}(xt^2)^n=\frac{1}{\sqrt{\pi}}\int_0^\infty dt\,e^{-t}t^{n-1/2}x^n=\frac{(n-\frac{1}{2})!}{(-\frac{1}{2})!}x^n\,, \tag{2.28}$$

which proves (2.25). Assume now that $0<\tau<\varrho$. Then by (2.17), (2.18), it follows that

$$\|\mathscr{B}g\|_\tau\leqq\text{const}\,\|g\|_\varrho\,. \tag{2.29}$$

---

[2] We prefer the notation $x!$ instead of $\Gamma(x+1)=\int_0^\infty t^x e^{-t}dt$ for the Euler gamma-function

Since polynomials are dense in $\mathscr{A}_\varrho$, it suffices to show (2.23) for $g(x)=x^n$. Assuming that $n>0$ (the case $n=0$ is trivial), the contour integral in (2.23) can be evaluated as follows.

$$\begin{aligned}\frac{1}{2\pi i}\oint_{|z|=\varrho}\frac{dz}{z}\exp\left(\frac{t}{(1+x)z}\right)z^{n-1}&=(1+x)^{1-n}\frac{1}{2\pi i}\oint_{|z|=\varrho}\frac{dz}{z}\exp\left(\frac{t}{z}\right)z^{n-1}\\&=(1+x)^{1-n}\sum_{k=0}^{\infty}\frac{t^k}{k!}\cdot\frac{1}{2\pi i}\oint_{|z|=\varrho}\frac{dz}{z}z^{n-k-1}\\&=(1+x)^{1-n}\frac{t^{n-1}}{(n-1)!}.\end{aligned}\tag{2.30}$$

Using this together with a standard formula [37] for the beta-function, the right-hand side of (2.23) becomes

$$\frac{t^n}{(n-1)!}\int_0^\infty\frac{dx}{x^{1/2}(1+x)^{n+1/2}}=\frac{(-\frac{1}{2})!}{(n-\frac{1}{2})!}t^n.\tag{2.31}$$

There remains (2.24) to be shown. But this estimate is now an immediate consequence of (2.23), since the exponential factor in the contour integral is bounded by $\exp(|t|/\varrho)$. □

This completes our discussion of the domain questions related to our version of the Borel transformation, and we start now the analysis of the renormalization transformation $\mathscr{T}$, as defined in (2.13). From the above discussion it follows that the map $\mathscr{T}$ is related to the map $\mathscr{T}_B$ by a "change of coordinates" in function space, namely

$$\mathscr{T}=\mathscr{B}^{-1}\mathscr{T}_B\mathscr{B}.\tag{2.32}$$

Proposition 2.9. *Let $\varrho=7/2$. Then the transformation $\mathscr{T}$ is a C^∞ map from $\mathscr{A}_\varrho^+$ to $\mathscr{A}_\varrho^+$, and*

$$(\mathscr{T}h)(x)=\tilde{h}(x)=\frac{1}{c}\cdot\frac{1}{\sqrt{1+\frac{x}{33}}}(\mathscr{B}^{-1}([\mathscr{B}h]^8))\left(\frac{\frac{3}{11}+\frac{x}{22}}{1+\frac{x}{33}}\right)\tag{2.33}$$

for h in $\mathscr{A}_\varrho^+$. Here, $c=c(h)$ is defined such that $\tilde{h}(0)=1$.

Proof. The regularity property of $\mathscr{T}$ could be proved directly by using (2.33). However we will not do this, since things already follow from Proposition 2.2 and Proposition 2.8. To show (2.33) we start with formula (2.1). Using (2.25) we get

$$\begin{aligned}\tilde{h}(x)\equiv(\mathscr{B}^{-1}\tilde{h}_B)(x)=\frac{1}{c}\cdot\frac{1}{\sqrt{\pi}}\int_{-\infty}^{\infty}dt\exp(-t^2)\exp\left(\left(\lambda-\frac{L^2\lambda}{1+\lambda l}\right)xt^2\right)\\\times\int_{-\infty}^{\infty}ds\exp(-(1+\lambda l)s^2)\left[h_B\left(\left(\frac{\alpha}{1+\lambda l}\sqrt{x}t+s\right)^2\right)\right]^l.\end{aligned}\tag{2.34}$$

Next we change variables from (t, s) to (t, τ), setting

$$s = s(t,\tau) \equiv q(x)\tau - \frac{\alpha}{1+\lambda l}\sqrt{x}t, \tag{2.35}$$

where

$$q(x) = \left(\frac{1+(\alpha^2-\lambda)x}{1+\lambda l+\lambda((\alpha^2-\lambda)l-1)x}\right)^{1/2}. \tag{2.36}$$

Then the integral with respect to dt in (2.34) can be computed explicitly, and we obtain

$$\tilde{h}(x) = \frac{1}{c}\cdot\frac{1}{\sqrt{1+\lambda l+\lambda((\alpha^2-\lambda)l-1)x}}\cdot(\mathscr{B}^{-1}([\mathscr{B}h]^l))(q(x)^2). \tag{2.37}$$

Equation (2.33) follows by inserting the numerical values given in (2.2) and (2.3). □

Theorem 2.4 is now a corollary of the following theorem for $\mathscr{T}$.

Theorem 2.10. *For $\varrho = 7/2$ there exists a fixed point h^* of $\mathscr{T}$ in $\mathscr{A}_\varrho^+$. The function $h_B^* = \mathscr{B}h^*$ is non-Gaussian.*

The proof of this theorem is outlined in the next section. The details are given in Sect. 5.

3. Proof

We prove Theorem 2.10 by reducing it to a large number of simple inequalities. We start by rewriting the fixed point problem for the RG transformation $\mathscr{T}$ as a fixed point problem for a contraction $\mathscr{M}$. $\mathscr{M}$ is then disassembled into a product of maps which can be bounded individually. This procedure leads to a very large number of inequalities which need to be checked, and we use at this point a computer to perform the bounds, and to assemble them properly.

3.1. The Contraction $\mathscr{M}$

In order to simplify some estimates, let us introduce the norm

$$\|h\| = \sum_{n=0}^{\infty}\frac{1}{n!}\left|\left(\left(\frac{d}{dz}\right)^n h\right)(0)\right|. \tag{3.1}$$

Furthermore, we denote by $\mathscr{A}$ the Banach space of functions h in $\mathscr{A}_1$ for which the norm (3.1) is finite. Instead of $\mathscr{T}$, we will now analyze the transformation

$$\mathscr{R} = \mathscr{J}_\varrho^{-1}\mathscr{T}\mathscr{J}_\varrho, \qquad \varrho = 7/2, \tag{3.2}$$

where $\mathscr{J}_r$ is defined for $r>0$ by

$$(\mathscr{J}_r h)(z) = h\left(\frac{z}{r}\right). \tag{3.3}$$

Since $\mathscr{A}$ contains $\mathscr{A}_\tau$ for every $\tau>1$, it follows from Proposition 2.2 and Proposition 2.8 that $\mathscr{R}$ is a C^∞ map on $\mathscr{A}\backslash\{0\}$.

We will now show how the fixed point problem for $\mathscr{R}$ can be reduced to a fixed point problem for a contraction. More precisely, we first define a map $\mathscr{M}$, which has a "good chance" of being a contraction, and whose fixed point (if it exists) is related to the fixed point of $\mathscr{R}$. The necessary properties of $\mathscr{M}$ are proved afterwards. A numerical analysis (whose details are irrelevant here) indicates that $\mathscr{R}$ has a nontrivial hyperbolic fixed point, with a one-dimensional local unstable manifold which is approximately parallel to the lines

$$\mathscr{L}_h = \{h + s \cdot e | s \in \mathbb{R}, e(z) = z + \tfrac{3}{5} z^2\} . \tag{3.4}$$

From these findings we can expect that, given a function h close to the fixed point of $\mathscr{R}$, there are points on $\mathscr{R}(\mathscr{L}_h)$ which are even closer. This motivates the following definition of $\mathscr{M}$. To simplify the notation, let us identify functions σ in $\mathscr{A}$ with sequences $(\sigma_2, \sigma_3, \ldots)$. The identification is made by means of the power series

$$\sigma(z) = \sum_{n=0}^{\infty} \sigma_{n+2} z^n , \qquad |z| \leqq 1 . \tag{3.5}$$

Consider the family $s \mapsto U_s$ of "unpack" operators on $\mathscr{A}$, given by

$$(U_s \sigma)(z) = 1 + az + (W_s \sigma)(z) , \tag{3.6}$$

$$(W_s \sigma)(z) = s \cdot e(z) + \sigma_2 z^2 + \tfrac{1}{3}\sigma_3 z^3 + \tfrac{1}{2}\sigma_4 z^4 + \sum_{n=5}^{\infty} \sigma_n z^n , \tag{3.7}$$

where $a = 0.6989740\ldots$ (see Sect. 5) and $e(z) = z + (3/5)z^2$ are fixed. Then we define $\mathscr{M}$ by the equation

$$U_s \mathscr{M} \sigma = \mathscr{R} U_s \sigma , \qquad s = s(\sigma) . \tag{3.8}$$

Here, $s = s(\sigma)$ is implicitly defined by the requirement that the right-hand side of (3.8) lies in the range of U_s (see Sect. 3.2 below). Note that if σ^* is a fixed point of $\mathscr{M}$ in $\mathscr{A}$, then

$$h^* = \mathscr{J}_\varrho^{-1} U_s \sigma^* , \qquad s = s(\sigma^*) , \tag{3.9}$$

is a fixed point of $\mathscr{T}$ in $\mathscr{A}_\varrho$.

The factors 3 and 2 in (3.7) have been chosen with the intention to minimize the norm of the tangent map of $\mathscr{M}$ near σ^*. Our next goal is to verify the conditions for the contraction mapping principle. This amounts to show that there is a function σ_0 in $\mathscr{A}$ such that

a) $\mathscr{M}$ is C^1 in a neighborhood $\mathscr{U}_\beta = \{\sigma \in \mathscr{A} | \, \|\sigma - \sigma_0\| < \beta\}$ of σ_0,

b) The tangent map $D\mathscr{M}_\sigma$ for $\sigma \in \mathscr{U}_\beta$ is bounded by $\|D\mathscr{M}_\sigma\| \leqq \theta < 1$,

c) σ_0 is an approximate fixed point of $\mathscr{M}$, i.e. $\|\mathscr{M}\sigma_0 - \sigma_0\| \leqq \varepsilon < \beta(1-\theta)$.

As a consequence of a), ..., c) there is a unique fixed point σ^* of $\mathscr{M}$ in $\mathscr{U}_\beta$, and Theorem 2.10 follows.

3.2. Formulas for $\mathscr{M}$ and $D\mathscr{M}$

We start the proof of a), ..., c) by replacing a) by a stronger but more convenient condition. Consider the function $s(\cdot)$ defined by (3.8). For given σ in $\mathscr{A}$, $s(\sigma)$ is obtained as the fixed point of the map

$$x \mapsto \tilde{s}(x, \sigma) \equiv (\mathscr{R} U_x \sigma)'(0) - a . \tag{3.10}$$

To find this fixed point we use Newton's method, i.e. we iterate the map

$$x \mapsto N_\sigma(x) = \left[\frac{\partial \tilde{s}}{\partial x}(x,\sigma) - 1\right]^{-1} (\tilde{s}(x,\sigma) - x), \tag{3.11}$$

starting with x in some given interval I. To guarantee convergence, it is sufficient to verify the conditions

$$\frac{\partial \tilde{s}}{\partial x}(x,\sigma) = (D\mathscr{R}_{U_{x\sigma}} e)'(0) > 2 \quad \text{and} \quad N_\sigma(x) \subset I, \tag{3.12}$$

for all pairs (x,σ) in $I \times \mathscr{U}_\beta$. Since $\tilde{s}$ is C^∞ on $I \times \mathscr{U}_\beta$, it follows that $s(\cdot)$, and thus also $\mathscr{M}$, are C^∞ on $\mathscr{U}_\beta$.

In the next subsection we will outline how the inequalities b), c) and (3.12) are proved on a computer. As a preparatory step we decompose $\mathscr{M}$ and $D\mathscr{M}$ into products of more elementary maps. First we have

$$\mathscr{M}\sigma = U_s^{-1} \mathscr{R} U_s \sigma, \tag{3.13}$$

where $s = s(\sigma)$ is the fixed point of (3.11). Note that all three factors in (3.13) are nonlinear. The nonlinear parts of $\mathscr{R}$ are given by

$$(Eh)(z) = [h(z)]^8, \tag{3.14}$$

and

$$(Sh)(z) = [h(0)]^{-1} h(z). \tag{3.15}$$

In order to be able to work with the spaces e_ϱ and $\mathscr{A}$ only, we combine E with a scaling $\mathscr{J}_r$ with $r = 8.956 > 8$. This leads to the following decomposition of $\mathscr{R}$.

$$\mathscr{R} = S\mathscr{K}, \qquad \mathscr{K} = QV\mathscr{B}^{-1} \mathscr{J}_r E \mathscr{B}. \tag{3.16}$$

Here, $\mathscr{B}$ is the Borel transform (2.21), and the remaining factors are defined by

$$(Vh)(z) = h\left(\frac{1}{\varrho \cdot r} \cdot \frac{\frac{3}{11} + \frac{\varrho}{22} z}{1 + \frac{\varrho}{33} z}\right), \tag{3.17}$$

and

$$(Qh)(z) = \left[1 + \frac{\varrho}{33} z\right]^{-1/2} h(z). \tag{3.18}$$

Using the definitions of $\mathscr{R}$ and U_s, we get the following expression for the tangent map $D\mathscr{M}$ of $\mathscr{M}$,

$$D\mathscr{M}_\sigma \delta\sigma = W_{\delta s}^{-1} D\mathscr{R}_{U_{s\sigma}} W_{\delta s} \delta\sigma, \tag{3.19}$$

where $s = s(\sigma)$, and where δs is given by

$$\delta s = \left[1 - \frac{\partial \tilde{s}}{\partial x}\right]^{-1} \left.\frac{\partial \tilde{s}}{\partial \sigma}\right|_{(s(\sigma),\sigma)} \delta\sigma. \tag{3.20}$$

The tangent map $D\mathscr{R}$ of $\mathscr{R}$, as obtained from (3.14), ..., (3.18), is

$$D\mathscr{R}_h = DS_{\mathscr{K}h} Q V \mathscr{B}^{-1} \mathscr{J}_r D E_{\mathscr{B}h} \mathscr{B}, \tag{3.21}$$

$$(DE_h \delta h)(z) = 8[h(z)]^7 \delta h(z), \tag{3.22}$$

$$(DS_h \delta h)(z) = [h(0)]^{-2}(h(0) \cdot \delta h(z) - h(z) \cdot \delta h(0)). \tag{3.23}$$

3.3. Bounds Used on the Computer

We first give a definition of what we call a bound in the context of computer assisted proofs.

Let Σ, Σ' be sets, and define $\mathscr{P}(\Sigma)$ and $\mathscr{P}(\Sigma')$ to be the set of all subsets of Σ and Σ' respectively. Let furthermore F and G be maps from $D_F \subset \mathscr{P}(\Sigma)$ and $D_G \subset \mathscr{P}(\Sigma)$ to $\mathscr{P}(\Sigma')$.

Definition 3.1. G is called a bound on F if

a) $D_F \supseteq D_G$,

b) $F(K) \subseteq G(K)$, $\forall K \in D_G$.

"G is a bound on F" will also be written as $G \geqq F$.

In this language, our estimate on the tangent map $D\mathscr{M}$ of $\mathscr{M}$ is a bound G on the map

$$F: K \mapsto \{\|D\mathscr{M}_\sigma\| \,|\sigma \in K\}, \qquad K \subset \mathscr{A}. \tag{3.24}$$

Bounds of the type just defined have some general properties which make it possible to "mechanize" estimates. One of them is the fact that if $G = G_1 \circ G_2$ is well defined, with $G_1 \geqq F_1$ and $G_2 \geqq F_2$, then G is a bound on $F_1 \circ F_2$. In order to ensure that bounds can be properly composed, we adapt the following procedure.

- Given Σ and Σ', we define a set of *standard sets*,

$$\mathrm{std}(\Sigma) \subset \mathscr{P}(\Sigma), \qquad \mathrm{std}(\Sigma') \subset \mathscr{P}(\Sigma'). \tag{3.25}$$

- Given a map

$$F: \mathscr{P}(\Sigma) \supseteq D_F \to \mathscr{P}(\Sigma'), \tag{3.26}$$

we then construct a map

$$G: \mathrm{std}(\Sigma) \supseteq D_G \to \mathrm{std}(\Sigma'), \tag{3.27}$$

such that $G \geqq F$.

Standard sets will be defined below for $\Sigma = \mathbb{R}$, $\mathscr{A}$, and e_ϱ. For products of such spaces we define

$$\mathrm{std}(\Sigma_1 \times \Sigma_2) = \mathrm{std}(\Sigma_1) \times \mathrm{std}(\Sigma_2). \tag{3.28}$$

Let us start with the bounds for binary operations on real numbers, such as "sum," "product," "equal," .... Each of these operations is regarded as a map from $\mathscr{P}(\mathbb{R} \times \mathbb{R})$ to $\mathscr{P}(\mathbb{R})$. For example

$$\mathrm{product}(K) = \{x \cdot y | (x, y) \in K\}, \qquad K \in \mathrm{std}(\mathbb{R} \times \mathbb{R}), \tag{3.29}$$

$$\mathrm{std}(\mathbb{R}) = \{[x^<, x^>] | x^<, x^> \in \mathfrak{R}; x^< \leqq x^>\}, \tag{3.30}$$

where $\mathfrak{R}$ is a fixed finite set of rational numbers, containing zero and one (see e.g. [1, 2, 24] for a discussion of such a set). Then for each of the maps F listed above, there is a bound

$$G: \mathrm{std}(\mathbb{R} \times \mathbb{R}) \supseteq D_G \to \mathrm{std}(\mathbb{R}) \tag{3.31}$$

with nonempty domain, and the graph of G is a finite set. It is clear that in general G is not unique. We have chosen $\mathfrak{R}$ and G in such a way that $G(K)$ can be evaluated on a computer. The details can be found in Sect. 5. Following the tradition of [1, 2], the elements of $\mathfrak{R}$ and of $\mathrm{std}(\mathbb{R})$ are referred to as *reps* and *scalars*, respectively. The bounds on "sum," "product," ..., are now regarded as given, and they will be denoted by "$s_$sum," "$s_$product," .... In order to make formulas more readable, we will also use notations like

$$\{x \cdot y\} \equiv s\_\mathrm{product}((x, y)) . \tag{3.32}$$

Having chosen all the necessary scalar bounds, we proceed by considering maps of the type (3.26), where Σ is a function space. A simple example of such a map would be

$$\mathrm{norm}(K) = \{\|h\| \,|h \in K\}, \qquad K \subset \mathscr{A} . \tag{3.33}$$

Following our standard procedure, we describe now first the standard sets for $\mathscr{A}$. We fix, once and for all, an integer $N > 0$. Then we define *vectors* as $(N+3)$-tuples $\mathbf{v} = (v_0, v_1, \ldots, v_N, v^{(G)}, v^{(H)})$ of scalars. To every vector $\mathbf{v}$ we associate a set

$$\hat{v} = \left\{ h \in \mathscr{A} \,|\, h(z) = \sum_{i=0}^{N} c_i z^i + z h^{(G)}(z) + z^{N+1} h^{(H)}(z);\ c_i \in v_i ; \right.$$
$$\left. h^{(G)} \in \mathscr{A},\ \|h^{(G)}\| \in v^{(G)};\ h^{(H)} \in \mathscr{A},\ \|h^{(H)}\| \in v^{(H)} \right\}, \tag{3.34}$$

and we choose $\mathrm{std}(\mathscr{A})$ to be the collection of all these sets $\hat{v}$. In order to simplify the notation, we shall only discuss here vectors $\mathbf{v}$ whose general components $v^{(G)}$ and higher order components $v^{(H)}$ are intervals which have zero as their lower boundary. Then the correspondence $\mathbf{v} \mapsto \hat{v}$ is one-to-one, and we can identify vectors and standard sets.

We are now in the position to give a bound on the norm (3.33). For example, if $s_$abs is a bound on the absolute value function, then the map

$$\mathbf{v} \mapsto s\_\mathrm{norm}(\mathbf{v}) = s\_\mathrm{abs}([s\_\mathrm{abs}([\ldots s\_\mathrm{abs}([s\_\mathrm{abs}([v_0 + v_1]) + v_2]) + \ldots]) + v^{(H)}]) \tag{3.35}$$

clearly satisfies all the necessary requirements. Similarly we can construct vector-valued bounds for elementary operations on $\mathscr{A}$ such as "sum," "product," .... For a descritpion of these bounds we refer to Sect. 5. A detailed discussion can also be found in [1–5, 7].

The bounds considered so far are the basic building blocks to prove Theorem 2.10. Such blocks can be assembled to provide bounds for the map

$$F: K \mapsto \{\|\mathscr{M}\sigma - \sigma\| \,|\sigma \in K\}, \qquad K \subset \mathscr{A}, \tag{3.36}$$

and for the map given in (3.24). Due to the large number of steps involved in this process, we have written this assembly plan in the form of a computer program.

The main part of this program is the equivalent of Sect. 3.2, with the original maps replaced by their corresponding bounds. A large portion of the other parts of the program contains the exact definitions of the general purpose bounds such as $s\_\mathrm{abs}, v\_\mathrm{add}, \ldots$. Finally, the program contains yet another part which is new and specific to our problem. Namely, we bound maps from and to the spaces e_ϱ. These bounds will be discussed in the next subsection.

To complete our discussion on bounds we define now the standard sets for the spaces e_ϱ [see (2.19) for the definition of e_ϱ]. We define a *Borel-function* to be a list

$$\mathbf{b}=(b_0, b_1, \ldots, b_N, b^{(G)}, b^{(H)}, b^{(E)}),$$

whose first $N+3$ components are scalars, and whose last component $b^{(E)}$ is a rep (i.e. an element of $\mathfrak{R}$). To every Borel-function $\mathbf{b}$ with $b^{(E)}=\varrho$, we associate a set

$$\hat{b}=\left\{h_B\in e_\varrho \,\middle|\, h_B(z)=\sum_{i=0}^{N} d_i z^i + z h_B^{(G)}(z) + z^{N+1} h_B^{(H)}(z);\ d_i\in b_i;\right.$$
$$\left. h_B^{(G)}\in \mathscr{E}_\varrho,\ \|h_B^{(G)}\|_\varrho \in b^{(G)};\ h_B^{(H)}\in \mathscr{E}_\varrho,\ \|h_B^{(H)}\|_\varrho \in b^{(H)}\right\}, \tag{3.37}$$

and we choose $\mathrm{std}(e_\varrho)$ to be the collection of all these sets $\hat{b}$. In order to simplify the notation, we shall only discuss Borel-functions $\mathbf{b}$ whose general components $b^{(G)}$ and higher order components $b^{(H)}$ are intervals which have zero as their lower boundary. Then the correspondence $\mathbf{b}\mapsto\hat{b}$ is one-to-one, and we can identify Borel-functions and standard sets.

3.4. Bounds on Borel-Functions

We construct here bounds for the following three maps.

a) The Borel transform

$$F:\mathscr{P}(\mathscr{A})\supseteq D_F\to\mathscr{P}(e_1),\qquad \mathbf{v}\mapsto\{\mathscr{B}h|h\in\mathbf{v}\}. \tag{3.38}$$

b) The inverse Borel transform

$$F:\mathscr{P}(e_\varrho)\supseteq D_F\to\mathscr{P}(\mathscr{A}),\qquad \mathbf{b}\mapsto\{(\mathscr{B}J_c)^{-1}h_B|h_B\in\mathbf{b}\}. \tag{3.39}$$

c) The product of Borel-functions

$$F:\mathscr{P}(e_{\varrho_1}\times e_{\varrho_2})\supseteq D_F\to\mathscr{P}(e_{\varrho_s}),\qquad K\mapsto\{g_{B,1}\cdot g_{B,2}|g_{B,1}\times g_{B,2}\in K\}. \tag{3.40}$$

In the construction of these bounds we will use a map $c\mapsto\{c\}$ which associates to every real number c, $|c|$ sufficiently small, a scalar $\{c\}$ which contains c. This allows us to define a product of real numbers with scalars, namely $(c,x)\mapsto\{c\cdot s\}\equiv\{\{c\}\cdot s\}$.

The Borel Transform. In order to bound the Borel transform (3.38), it suffices to construct a map

$$G:\mathrm{std}(\mathscr{A})\supseteq D_G\to\mathrm{std}(e_1), \tag{3.41}$$

with the property that for all vectors $\mathbf{v}\in D_G$,

$$\{\mathscr{B}h|h\in\mathbf{v}\}\subseteq G(\mathbf{v}). \tag{3.42}$$

Let now $\mathbf{v}=(v_0, v_1, \ldots, v_N, v^{(G)}, v^{(H)})$ be a vector. Then we define a Borel-function

$$\mathbf{b}=(b_0, b_1, \ldots, b_N, b^{(G)}, b^{(H)}, 1)$$

in $\mathrm{std}(e_1)$ by

$$b_i=\left\{\frac{(-\frac{1}{2})!}{(i-\frac{1}{2})!}\cdot v_i\right\}, \tag{3.43}$$

$$b^{(G)}=[0, b_G^{>}]\cdot \quad b_G^{>}=\left\{\frac{(-\frac{1}{2})!}{(\frac{1}{2})!}\cdot v^{(G)}\right\}^{>}, \tag{3.44}$$

$$b^{(H)}=[0, b_H^{>}], \quad b_H^{>}=\left\{\frac{(-\frac{1}{2})!}{(N+\frac{1}{2})!}\cdot v^{(H)}\right\}^{>}, \tag{3.45}$$

where $\{\ \}^{>}$ denotes the upper boundary of the scalar $\{\ \}$. We claim that the map $G: \mathbf{v}\mapsto\mathbf{b}$ satisfies (3.42). It is clear that this is the case for polynomials, i.e. when $v^{(G)}$ and $v^{(H)}$ are zero. It is also easy to see, that it then suffices to restrict the analysis to functions

$$h(z)=z^k\cdot l(z), \qquad l\in\mathscr{A}, \tag{3.46}$$

for $k=1$ or $k=N+1$. We will show that for $t\in\mathbb{C}$

$$\left|t^{-k}(\mathscr{B}h)(t)\right|\leqq\frac{(-\frac{1}{2})!}{(k-\frac{1}{2})!}\exp(|t|), \tag{3.47}$$

from which (3.42) follows. To prove (3.47), we use the integral representation (2.23) for the Borel transform $\mathscr{B}$. We have

$$\begin{aligned}
h_B(t)=(\mathscr{B}h)(t)&=t\cdot\int_0^\infty\frac{dx}{(1+x)^{3/2}\sqrt{x}}\left[\frac{1}{2\pi i}\oint_{|z|=1}\frac{dz}{z}\exp\left(\frac{t}{z(1+x)}\right)\frac{h(z)}{z}\right]\\
&=t\cdot\int_0^\infty\frac{dx}{(1+x)^{3/2}\sqrt{x}}\left[\frac{1}{2\pi i}\oint_{|z|=1}\frac{dz}{z}\exp\left(\frac{t}{z(1+x)}\right)z^{k-1}l(z)\right]\\
&=t^k\cdot\int_0^\infty\frac{dx}{(1+x)^{k+1/2}\sqrt{x}}\left[\frac{1}{2\pi i}\oint_{|z|=1}\frac{dz}{z}\cdot\frac{\exp\left(\frac{t}{z(1+x)}\right)-\sum_{\nu=0}^{k-2}\frac{1}{\nu!}\left(\frac{t}{z(1+x)}\right)^\nu}{\left(\frac{t}{z(1+x)}\right)^{k-1}}l(z)\right]
\end{aligned} \tag{3.48}$$

The contour integral is bounded by using

$$\begin{aligned}
\left|\frac{e^\xi-\sum_{\nu=0}^{k-2}\frac{\xi^\nu}{\nu!}}{\xi^{\nu-1}}\right|&=\left|\sum_{\nu=k-1}^\infty\frac{\xi^{\nu-k+1}}{\nu!}\right|=\left|\sum_{\nu=0}^\infty\frac{\xi^\nu}{(\nu+k-1)!}\right|\\
&=\frac{1}{(k-1)!}\left|\sum_{\nu=0}^\infty\frac{(k-1)!\nu!}{(\nu+k-1)!}\cdot\frac{\xi^\nu}{\nu!}\right|\leqq\frac{1}{(k-1)!}\exp(|\xi|),
\end{aligned} \tag{3.49}$$

and therefore, using a standard integral representation for the beta-function [37], we get

$$|t^{-k}(\mathscr{B}h)(t)| \leqq \int_0^\infty \frac{dx}{(1+x)^{k+1/2}\sqrt{x}} \cdot \frac{1}{2\pi} \int_0^{2\pi} \frac{d\varphi}{(k-1)!} \exp\left(\frac{|t|}{1+x}\right) \|l\|_1$$

$$\leqq \frac{(-\frac{1}{2})!}{(k-\frac{1}{2})!} \|l\|_1 \exp(|t|). \tag{3.50}$$

This proves (3.47).

The Inverse Borel Transform. We bound the map (3.39) for $c > \varrho$ and $\varrho \in \mathfrak{R}$ by constructing a map

$$G: \mathrm{std}(e_\varrho) \supseteq D_G \to \mathrm{std}(\mathscr{A}), \tag{3.51}$$

which satisfies

$$\{(\mathscr{B}\mathscr{J}_c)^{-1} h_B | h_B \in \mathbf{b}\} \subseteq G(\mathbf{b}), \qquad \mathbf{b} \in D_G. \tag{3.52}$$

Given a Borel-function $\mathbf{b} = (b_0, b_1, \ldots, b_N, b^{(G)}, b^{(H)}, \varrho)$, we define a vector

$$\mathbf{v} = (v_0, v_1, \ldots, v_N, v^{(G)}, v^{(H)})$$

by choosing

$$v_i = \left\{\frac{(i-\frac{1}{2})!}{(-\frac{1}{2})!} \cdot b_i \cdot c^i\right\}, \tag{3.53}$$

$$v^{(G)} = [0, v_G^>], \qquad v_G^> = \left\{(b^{(G)})^> \frac{(\frac{1}{2})!}{(-\frac{1}{2})!} \cdot \frac{1}{(1-\tau/\varrho)^{3/2}} \cdot \frac{1}{(1-c/\tau)}\right\}^>, \tag{3.54}$$

$$v^{(H)} = [0, v_H^>], \qquad v_H^> = \left\{(b^{(H)})^> \frac{(N-\frac{1}{2})!}{(-\frac{1}{2})!} \cdot \frac{c^{N+1}}{(1-\tau/\varrho)^{N+3/2}} \cdot \frac{1}{(1-c/\tau)}\right\}, \tag{3.55}$$

where $\tau = (2/5)\varrho + (3/5)c$. We claim that the map $G: \mathbf{b} \mapsto \mathbf{v}$ satisfies (3.52). As above, the "polynomial part" (3.53) is trivial, so we may assume that $\mathbf{b}$ only contains functions of the form

$$h_B(t) = t^k l_B(t), \qquad l_B \in \mathscr{E}_\varrho, \tag{3.56}$$

with $k = 1$ or $k = N+1$. The inverse Borel transform of such a function h_B is given by

$$h(z) = (\mathscr{B}^{-1} h_B)(z) = \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} dt \exp(-t^2) h_B(zt^2) = \frac{z^k}{\sqrt{\pi}} \int_{-\infty}^{\infty} dt \exp(-t^2) t^{2k} l_B(zt^2). \tag{3.57}$$

Therefore

$$|h(z)| \leqq \frac{|z|^k}{\sqrt{\pi}} \int_{-\infty}^{\infty} dt \exp(-t^2) t^{2k} \|l_B\|_\varrho \exp\left(\frac{|z|t^2}{\varrho}\right), \tag{3.58}$$

and

$$\sup_{|z|\leqq 1} |\mathscr{J}_\tau^{-1} h(z)| = \sup_{|z|\leqq 1} |h(\tau z)| \leqq \frac{\tau^k}{\sqrt{\pi}} \left(\int_{-\infty}^{\infty} t^{2k} \exp(-t^2(1-\tau/\varrho))dt \right) \|l_B\|_\varrho$$
$$= \frac{(k-\frac{1}{2})!}{(-\frac{1}{2})!} \cdot \frac{\tau^k}{(1-\tau/\varrho)^{k+1/2}} \cdot \|l_B\|_\varrho . \tag{3.59}$$

To bound the norm of $\mathscr{J}_\tau^{-1} h$ as an element of $\mathscr{A}$ [see (3.1) for the definition of $\mathscr{A}$] we use Cauchy's formula with contour $|z| = \tau/\varrho$, and obtain

$$\|(\mathscr{B}\mathscr{J}_c)^{-1} h_B\| = \|\mathscr{J}_c^{-1} h\| \leqq \frac{(c/\tau)^k}{1-(c/\tau)} \sup_{|z|=1} |h(\tau z)|$$
$$\leqq \frac{(k-\frac{1}{2})!}{(-\frac{1}{2})!} \cdot \frac{c^k}{(1-\tau/\varrho)^{k+1/2}} \cdot \frac{1}{1-c/\tau} \cdot \|l_B\|_\varrho . \tag{3.60}$$

This establishes (3.52) for the bound G, defined by (3.53), ..., (3.55).

Remark. The particular choice $\tau = (2/5)\varrho + (3/5)c$ has been made in order to minimize the right-hand side of (3.60) for $k=1$. For large k, (3.60) yields a very poor bound even for optimal choices of τ. We follow therefore the strategy to transform "higher order bounds" into "lower order bounds" whenever this is possible without negative consequences for subsequent estimates. More details can be found below in the discussion of the product of Borel-functions.

The Product of Borel-Functions. Given $\varrho_1, \varrho_2, \varrho_3 \in \mathfrak{R}$, satisfying $\varrho_3 > \varrho_1 + \varrho_2$, we construct a bound

$$G : \mathrm{std}(e_{\varrho_1} \times e_{\varrho_2}) \supseteq D_G \to \mathrm{std}(e_{\varrho_3}) \tag{3.61}$$

on the product (3.40), satisfying

$$\{h_{B,1} \cdot h_{B,2} | h_{B,1} \in \mathbf{b}_1, h_{B,2} \in \mathbf{b}_2\} \subseteq G((\mathbf{b}_1, \mathbf{b}_2)) . \tag{3.62}$$

Given two Borel-functions $\mathbf{b}_1$ and $\mathbf{b}_2$ in e_{ϱ_1} and e_{ϱ_2} respectively, we define $G((\mathbf{b}_1, \mathbf{b}_2)) = \mathbf{b}_3$, where

$$b_{3,i} = \left\{ \sum_{k=0}^{i} b_{1,k} b_{2,i-k} \right\}, \qquad i = 0, \ldots, N, \tag{3.63}$$

$$b_3^{(G)} = [0, (b_3^{(G)})^{>}], \tag{3.64}$$

$$b_3^{(H)} = [0, 0], \tag{3.65}$$

with $(b_3^{(G)})^{>}$ to be specified below. The sum in (3.63) is defined by repeated application of the scalar bound "s_sum." For the precise order of summation see Sect. 5. Since as noted above, the inverse Borel transform (3.53), ..., (3.55) gives poor estimates on higher order terms, we have chosen here to add all of these terms to the general bound $b_3^{(G)}$. This can be done by using the inequality

$$x^k \leqq \left(\frac{k}{e}\right)^k \cdot \exp(x), \qquad k \in \mathbb{N},\ x \in \mathbb{R} . \tag{3.66}$$

We define, for $\varepsilon = \{\varrho_3 - (\varrho_1 + \varrho_2)\}^>$

$$
\begin{aligned}
(b_3^{(G)})^> = \Bigg\{ & \left(\frac{1}{\varepsilon e}\right) b_1^{(G)} b_2^{(G)} + \left(\frac{2N+1}{\varepsilon e}\right)^{2N+1} b_1^{(H)} b_2^{(H)} \\
& + \sum_{i=N+1}^{2N} \left| \sum_{k=i-N}^{N} b_{1,k} b_{2,i-k} \right| \left(\frac{i-1}{e(\varrho_1+\varrho_2+\varepsilon)}\right)^{i-1} \\
& + \sum_{i=0}^{N} |b_{1,i}| b_2^{(G)} \left(\frac{i}{(\varrho_1+\varepsilon)e}\right)^{i} + \sum_{i=0}^{N} |b_{2,i}| b_1^{(G)} \left(\frac{i}{(\varrho_2+\varepsilon)e}\right)^{i} \\
& + \sum_{i=0}^{N} |b_{1,i}| b_2^{(H)} \left(\frac{N+i}{(\varrho_1+\varepsilon)e}\right)^{N+i} + \sum_{i=0}^{N} |b_{2,i}| b_1^{(H)} \left(\frac{N+i}{(\varrho_2+\varepsilon)e}\right)^{N+i} \Bigg\}^> . \qquad (3.67)
\end{aligned}
$$

Again, the sums and products in (3.67) are to be replaced by their corresponding scalar bounds. Similarly $s \mapsto |s|$ is a shorthand notation for the bound on the absolute value function. It is now clear that the map $G: (\mathbf{b}_1, \mathbf{b}_2) \mapsto \mathbf{b}_3$ satisfies (3.62), and is therefore a bound on the product (3.40).

Remark. The domain D_G of G is defined in terms of the domains of the maps "$s_$sum," "$s_$product," .... Namely, D_G is restricted to pairs $(\mathbf{b}_1, \mathbf{b}_2)$, for which the scalar bounds in (3.63) and (3.67) can be evaluated. Similar restrictions apply to the other bounds discussed in this subsection. Their exact definitions can be found in the computer program in Sect. 5. Such restrictions are implemented there in the form of "conditional statements," which result in error messages in the case of a domain violation.

4. Infinite Volume Limit and Correlations

In this section we construct the infinite volume measure corresponding to the function h_B^* described in Theorem 2.4, and discuss its two point function.

Denote by f^* the function defined by the equation

$$f^*(t) = \text{const} \exp(-\tfrac{1}{3}t) h_B^*(t), \qquad (4.1)$$

where h_B^* is the fixed point of $\mathcal{T}_B$. Since h_B^* is a solution of Eq. (2.1) for $\lambda = 1/3$, f^* solves Eq. (1.19), for $\lambda = 0$, and for some choice of the constant in (4.1). From Theorem 2.4 it follows that f^* satisfies the bound

$$|f^*(t)| \leqq \text{const} \exp(-\tfrac{1}{3}\operatorname{Re} t + \tfrac{2}{7}|t|), \qquad (4.2)$$

and therefore f^* is exponentially decreasing along the positive real axis. A repeated application of the identity (1.19) for f^* leads to a bound

$$\frac{1}{k(\delta)} \exp(-(\tfrac{3}{8}+\delta)t^2) < f^*(t^2) < k(\delta) \exp(-(\tfrac{3}{8}-\delta)t^2), \qquad t \in \mathbb{R}, \qquad (4.3)$$

for every $\delta > 0$, $0 < k(\delta) < \infty$.

We consider now finite volume measures $d\mu^{(n)}$ associated with a sequence of cubes $\Lambda_1, \Lambda_2, \ldots$

$$\Lambda_n = \{-L^n, \ldots, L^n - 1\}^3 . \qquad (4.4)$$

If ϕ is a spin configuration on Λ_n, then we define $d\mu^{(n)}$ by the equation

$$d\mu^{(n)}(\phi) = d\mu_{C_n}(\phi) F_n(\phi), \tag{4.5}$$

where

$$F_n(\phi) = \frac{1}{Z_n} \cdot \prod_{j \in \Lambda_n} f^*(((A^* A\phi)_j)^2), \tag{4.6}$$

and where

$$C_n = \frac{l}{2}\left(1 + L^{-2} \sum_{k=2}^{n} L^{2k} (A^*)^k A^k\right). \tag{4.7}$$

The normalization constants Z_n are chosen such that the measures $d\mu^{(n)}$ are probability measures. C_n is the hierarchical covariance introduced in (1.10), restricted to the cube Λ_n.

Proposition 4.1. *For $n = 2, 3, \ldots$ the covariances C_n and the functions F_n satisfy the following recursion relations*

$$C_n = \alpha^2 B^* C_{n-1} B + \Gamma, \tag{4.8}$$

$$F_{n-1}(\phi) = \widetilde{F}_n(\phi) \equiv \int d\mu_\Gamma(\psi) F_n(\alpha B^* \phi + \psi), \tag{4.9}$$

with $B^* = L^{d/2}(A^*)^2 A$ and $\Gamma = l/2 \cdot \mathrm{Id}$, as defined in Sect. 1. Furthermore the normalization constants Z_n satisfy $Z_n = Z_1$ for all n.

Proof. The recursion relation (4.8) is an immediate consequence of the definition (4.7). In order to prove (4.9) we compute $\widetilde{F}_n$,

$$\widetilde{F}_n(\phi) = \frac{1}{Z_n} \int_{-\infty}^{\infty} \left(\prod_{j \in \Lambda_n} \frac{d\psi_j}{\sqrt{\pi l}}\right) \exp\left(-\tfrac{1}{2}\left\langle \psi, \frac{2}{l}\psi \right\rangle\right) \prod_{j \in \Lambda_n} f^*(((A^* A(\alpha B^* \phi + \psi))_j)^2). \tag{4.10}$$

Since the integrand only depends on the block part $s_{[L^{-1}j]} = (A^* A\psi)_j$ of ψ_j, the fluctuations $(1 - A^* A)\psi$ can be integrated out, and therefore

$$\begin{aligned}
\widetilde{F}_n(\phi) &= \frac{1}{Z_n} \prod_{i \in \Lambda_{n-1}} \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} ds_i \exp(-s_i^2) \prod_{j:[L^{-1}j]=i} f^*(((L(A^*)^2 A\phi)_j + s_i)^2) \\
&= \frac{1}{Z_n} \prod_{i \in \Lambda_{n-1}} \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} ds_i \exp(-s_i^2) [f^*((\alpha(A^* A\phi)_i + s_i)^2)]^l \\
&= \frac{1}{Z_n} \prod_{i \in \Lambda_{n-1}} f^*(((A^* A\phi)_i)^2) = \frac{Z_{n-1}}{Z_n} F_{n-1}(\phi).
\end{aligned} \tag{4.11}$$

There remains to be shown that $Z_n = Z_{n-1}$. Using (4.8), we get

$$\begin{aligned}
1 &= \int d\mu_{C_n}(\phi) F_n(\phi) = \int d\mu_{C_{n-1}}(\phi) \int d\mu_\Gamma(\psi) F_n(\alpha B^* \phi + \psi) \\
&= \int d\mu_{C_{n-1}}(\phi) \frac{Z_{n-1}}{Z_n} F_{n-1}(\phi) = \frac{Z_{n-1}}{Z_n}.
\end{aligned} \tag{4.12}$$

This completes the proof of Proposition 4.1. □

In the following we discuss the thermodynamic limit of the measures $d\mu^{(n)}$. It is convenient to do this by considering the corresponding characteristic functionals $\mathcal{S}_n$. The characteristic functional, or Fourier transform, of the measure $d\mu^{(n)}$, is defined by the equation

$$\mathcal{S}_n\{g\} = \int d\mu^{(n)}(\phi) \exp(i\langle \phi, g\rangle), \tag{4.13}$$

where $\langle , \rangle$ denotes the inner product in $l_2(\mathbb{Z}^3)$.

The limit $n \to \infty$ will be performed by using the following properties.

Proposition 4.2. *Let g be a complex valued function on Λ_m and let $\beta > |g_j|$ for all $j \in \Lambda_m$. Then for $n \geqq m$*

$$|\mathcal{S}_n\{g\}| \leqq \exp(s(g) + \kappa \cdot 8^{m-n}), \tag{4.14}$$

where κ is some constant independent of m, n, and g, and where

$$s(g) = \kappa \cdot \sum_{j \in \Lambda_n} [(\beta^{4/3} + \beta^{-2/3})|g_j|^{2/3} + |g_j|^2]. \tag{4.15}$$

Furthermore the characteristic functionals $\mathcal{S}_n$ are entire analytic in all of their arguments g_j, and they satisfy $\mathcal{S}_n\{0\} = 1$.

Proof. The characteristic functional $\mathcal{S}_n$ can be written as

$$\begin{aligned} \mathcal{S}_n\{g\} = \frac{1}{Z_1} \int d\mu_{C_n}(\phi) \exp(i\langle (1 - A^*A)\phi, g\rangle) \\ \times \prod_{j \in \Lambda_n} f^*(((A^*A\phi)_j^2) \exp(i \cdot g_j (A^*A\phi)_j). \end{aligned} \tag{4.16}$$

As in (4.10) the integral over the fluctuations $r = (1 - A^*A)\phi$ factorizes. It can be computed explicitly, namely

$$\begin{aligned} |\int d\mu_{C_n}(r) \exp(i\langle r, (1 - A^*A)g\rangle)| &= |\exp(-\tfrac{1}{2}\langle (1 - A^*A)g, C_n(1 - A^*A)g\rangle)| \\ &\leqq \exp(2\langle g, g\rangle). \end{aligned} \tag{4.17}$$

The remaining terms will be bounded by using the exponential decay of the function f^*. From (4.3) it follows that there are constants k and κ, such that for $0 < p < 1$

$$\begin{aligned} f(t^2) \exp(|\sigma t|) &\leqq (f(t^2))^{1-p} (k \exp(-t^2/4))^p \exp(|\sigma t|) \\ &\leqq (f(t^2))^{1-p} \exp(4 \cdot |\sigma/p|^2 + p \cdot \ln k) \\ &\leqq (f(t^2))^{1-p} \exp(\kappa(\beta^{4/3} + \beta^{-2/3})\sigma^{2/3}). \end{aligned} \tag{4.18}$$

The last inequality is obtained by taking $p = (\sigma/\beta)^{2/3} < 1$. This bound can be applied to each factor in (4.16) separately, yielding

$$|\mathcal{S}_n\{g\}| \leqq \frac{1}{Z_1} \exp(s(g)) \int d\mu_{C_n}(\phi) \prod_{j \in \Lambda_n} [f(((A^*A\phi)_j)^2)]^{1-p_j}, \tag{4.19}$$

where $s(g)$ is given by (4.15), and where

$$p_j = (|g_j|/\beta)^{2/3} < 1. \tag{4.20}$$

Denote by F the product in (4.19). Using the relation (4.8) the integral in (4.19) can be written as the integral of $d\mu_{C_{n-1}}\tilde{F}$, with

$$\tilde{F}(\phi)=\int d\mu_T(\psi)F(\alpha B^*\phi+\psi)\,. \tag{4.21}$$

The right-hand side of (4.21) factorizes into one-dimensional integrals, which can be bounded by using the fact that f^* is a solution of Eq. (1.19) for $\lambda=0$. Applying Hölders inequality we get

$$\begin{aligned}\tilde{F}(\phi)&=\prod_{i\in\Lambda_{n-1}}\frac{1}{\sqrt{\pi}}\int_{-\infty}^{\infty}ds_i\exp(-s_i^2)\,[f((\alpha(A^*A\phi)_i+s_i)^2)]^{8(1-\tilde{p}_i)}\\&\leqq\prod_{i\in\Lambda_{n-1}}[f(((A^*A\phi)_i)^2)]^{1-\tilde{p}_i},\end{aligned} \tag{4.22}$$

where

$$\tilde{p}_i=\tfrac{1}{8}\sum_{j:[L^{-1}j]=i}p_j\,. \tag{4.23}$$

Integrating this bound on $\tilde{F}$, we get

$$|\mathscr{S}_n\{g\}|\leqq\frac{1}{Z_1}\exp(s(g))\int d\mu_{C_{n-1}}(\phi)\prod_{i\in\Lambda_{n-1}}[f(((A^*A\phi)_i)^2)]^{1-\tilde{p}_i}\,. \tag{4.24}$$

The steps (4.19) to (4.24) can now be iterated $n-1$ times. The integral which we obtain that way is bounded by $Z_1^{1-a_n}$, where a_n is the average value of the function $j\mapsto p_j$ on the cube Λ_n. This proves the bound (4.14). The analyticity of the functionals $\mathscr{S}_n$ follows from the absolute convergence of the integral (4.13). This completes the proof of Proposition 4.2. □

Corollary 4.3. *There exists a thermodynamic limit measure $d\mu$ for the sequence of finite volume measures $d\mu^{(n)}$. The Fourier transform $\mathscr{S}$,*

$$\mathscr{S}\{g\}=\int d\mu(\phi)\exp(i\langle\phi,g\rangle)\,, \tag{4.25}$$

is analytic in all g_j, and it satisfies the bound

$$|\mathscr{S}_n\{g\}|\leqq\exp(s(g))\,, \tag{4.26}$$

with $s(g)$ given by (4.15).

To be more specific, $\mathscr{S}$ is defined on the space $\mathscr{D}$ of functions g on $\mathbb{Z}^3$ with compact support, and is obtained as the limit of a subsequence of $(\mathscr{S}_1,\mathscr{S}_2,\ldots)$. The limit is constructed by using Montel's theorem. It then follows (see e.g. [21]) that $\mathscr{S}$ is the Fourier transform of a probability measure $d\mu$ on the (algebraic) dual $\mathscr{D}^*$ of $\mathscr{D}$.

In the remaining part of this section we analyze the second moment of the infinite volume measure $d\mu$. Under an assumption on the tangent map of the RG transformation $\mathscr{T}_B$, we will show that the long distance behavior of the two point correlation function is canonical, i.e. that the critical index η, as defined through Eq. (1.1), is zero in our model.

The moments of $d\mu$ are obtained by differentiating the characteristic functional (4.25) with respect to the parameters g_j. Note that by Cauchy's theorem, the

moments of the finite volume measures $d\mu^{(n)}$ converge to the moments of $d\mu$, along the same sequence Ω of indices as used in the construction of $\mathscr{S}$. Since the fluctuations $(1-A^*A)g$ contribute only a trivial factor to (4.25), we shall assume that $(1-A^*A)g=0$. Then $\mathscr{S}\{g\}$ can be written as a partition function

$$\mathscr{Z}(\mathbf{f})=\lim_{n\in\Omega}\frac{1}{Z_n}\int d\mu_{C_n}(\phi)\prod_{j\in\Lambda_n} f_j((A^*A\phi)_j), \tag{4.27}$$

with a single spin distribution $\mathbf{f}: j\mapsto f_j$ given by

$$f_j(t)=f^*(t^2)\exp(i\cdot tg_j). \tag{4.28}$$

Following the RG philosopy, we study the long distance behavior of the measure $d\mu$ by successively integrating out small scale degrees of freedom in (4.27). From the identity (1.6), we obtain

$$\mathscr{Z}(\mathbf{f})=\mathscr{Z}(\hat{\mathbf{f}}), \tag{4.29}$$

$$\hat{f}_i(t)=\frac{1}{\sqrt{\pi}}\int_{-\infty}^{\infty} ds\exp(-s^2)\prod_{j:[L^{-1}j]=i} f_j(\alpha t+s). \tag{4.30}$$

Note that the moments of $d\mu$ can also be written in the form (4.27), if we allow some of the functions f_j to be replaced by the corresponding g_j-derivatives. We shall only consider here the two point functions

$$S_2(k,m;h)=\lim_{n\in\Omega}\frac{1}{Z_n}\int d\mu_{C_n}(\phi)\prod_{j\in\Lambda_n\setminus\{k,m\}} f^*(((A^*A\phi)_j)^2)\prod_{j\in\{k,m\}} h((A^*A\phi)_j). \tag{4.31}$$

In this particular case, and for $|k-m|>3$, the identity (4.29) reduces to

$$S_2(k,m;h)=S_2\left([L^{-1}k],[L^{-1}m];\frac{1}{l}Th\right), \tag{4.32}$$

with T defined by the equation

$$(Th)(t)=l\cdot\frac{1}{\sqrt{\pi}}\int_{-\infty}^{\infty} ds\exp(-s^2)[f^*((\alpha t+s)^2)]^{l-1}h(\alpha t+s). \tag{4.33}$$

We choose now, once and for all, the function h to be

$$h(t)=t\cdot f^*(t^2). \tag{4.34}$$

For this choice of h the two point function (4.31) becomes

$$S_2(k,m;h)=\int d\mu(\phi)(A^*A\phi)_k(A^*A\phi)_m, \tag{4.35}$$

and we can apply the identity (4.32) to estimate its long distance behavior. The functions $Th, T^2h, \ldots$ can be computed explicitly. The result is

$$(T^nh)(t)=(\alpha l)^nh^{(n)}(t)\equiv(\alpha l)^n\cdot\frac{l}{L^2-1}\cdot[h_2(t)-L^{-2n}h_1(t)], \tag{4.36}$$

where

$$h_1(t)=t\cdot(f^*)'(t^2),\qquad h_2(t)=\frac{L^2-1}{l}\cdot t\cdot f^*(t^2)+h_1(t). \tag{4.37}$$

Remark. The map T is related to an extension of the RG transformation $\mathscr{T}_B$. This extension has two additional eigenvalues of modulus >1, namely αl and $\alpha l/L^2$. The functions h_1 and h_2 are the corresponding eigenvectors for T.

We now iterate the relation (4.32) as many times as possible. The maximum number of iterations depends on the points $k, m\in\mathbb{Z}^3$, and is given by

$$n_0=\max\{n|[L^{-n}k]\neq[L^{-n}m]\}=\frac{\log|k-m|}{\log L}+\mathcal{O}\left(\frac{1}{|k-m|}\right). \tag{4.38}$$

This leads to the following bound on the two point function (4.35),

$$\begin{aligned}S_2(k,m;h)&=S_2\left([L^{-n_0}k],[L^{-n_0}m];\left(\frac{1}{l}T\right)^{n_0}h\right)\\&=\alpha^{2n_0}S_2([L^{-n_0}k],[L^{-n_0}m];h^{(n_0)})\\&=\alpha^{2n_0}\left(\frac{l}{L^2-1}\cdot S_2([L^{-n_0}k],[L^{-n_0}m];h_2)-\mathcal{O}(L^{-2n_0})\right)\\&=\frac{Z}{|k-m|^{d-2}}+\mathcal{O}\left(\frac{1}{|k-m|^{d-1}}\right).\end{aligned} \tag{4.39}$$

where Z is some nonnegative constant. Here, we have used that there are three constants X, Y, Z, such that

$$S_2([L^{-n_0}k],[L^{-n_0}m];\varepsilon h_1+h_2)=X\varepsilon^2+Y\varepsilon+Z, \tag{4.40}$$

for every ε and for every choice of (k, m, n_0) satisfying (4.38). This can be seen by iterating the RG transformation defined in (4.29). For the first step, the integrals (4.30) are given by

$$\hat{f}_i(t)=\frac{1}{\sqrt{\pi}}\int_{-\infty}^{\infty}ds\exp(-s^2)[f^*((\alpha t+s)^2)]^{l-p}[(\varepsilon h_1+h_2)(\alpha t+s)]^p, \tag{4.41}$$

with $p=2$ if $i=[L^{-n_0-1}k]=[L^{-n_0-1}m]$, and $p=0$ in all other cases. This shows that the right-hand side of (4.40) is independent of k, m, and n_0. To bound the constants X, Y, and Z, we can follow the proof of Proposition 4.2.

Remark. This does not prove that the critical index η as defined in (1.1) is zero, since the field strength Z in (4.39) might vanish. To eliminate this possibility, we would have to analyze the map T (or the tangent map of $\mathscr{T}_B$) in more detail. A sufficient result would be that the function g, defined by

$$g(t)=[f^*(t^2)]^{-1}[h_2(t)]^2, \tag{4.42}$$

has a nonzero component in the spectral subspace associated with the eigenvalue 8 of T.

5. The Computer Program

```
/* 5.1. DEFINITIONS, TYPES, DECLARATIONS */
/* ================================== */

#define degree_plus_one 56
#define number_of_calculated_column_vectors 6
#define radius_of_domain 3.5e+00
#define radius_of_ball 5.0e-09
#define scaling_factor 8.956e+00
#define approx_linear_coeff 6.989740395952895e-01

/* types and external variables */
#include "Section5.2"

/* reps-macros and rup */
#include "Section5.8"

/* More declarations */

main()
{
 extern int
   n,n1;

 extern reps
   rrho,rup(),rmaxabs();

 extern scalar
   srho;

 extern vector
   varg,vfact;

 int
   j,jmax;

 reps
   repsilon,rbeta,rtheta;

 scalar st1;

 vector
   v0,v0i,v0t1,v0t2;

/* Initialisations */

 n1=degree_plus_one,
 n=n1-1;
 read_approx_fixed_point(&v0);
 rrho=radius_of_domain;
 srconst(rrho,&srho);
 init_pack_and_unpack();
 init_multiplication_by(&vfact);
 init_composition_with(&varg);

/* The approximate fixed point */

/* calculate image */
 m(&v0,&v0i),

/* calculate norm of difference */
 printf("NORM OF DIFFERENCE .\n\n");
 vequal(&v0i,&v0t1);
 vminus(&v0,&v0t1);
 snorm(&v0t1,&st1);
 repsilon=rmaxabs(&st1);
 printf("epsilon %32.16e %30.16e\n\n",st1.l,st1.u),

/* Bound on the tangent map DM */

/* prepare ball around approx fixed point */
 rbeta=radius_of_ball;
 sscalar(rzero,rbeta,&v0.g);
 m(&v0,&v0t1);

/* norm of polynomial part */
 printf("BOUND ON DM .\n\n");
 jmax=number_of_calculated_column_vectors;
 rtheta=rzero;
 for(j=2;j<=jmax;j++){
   vsetzo(&v0t1);
   sequal(&sone,v0t1.p+j);
   dm(&v0t1,&v0t2);
   snorm(&v0t2,&st1);
   printf("column %d %31.16e %30.16e\n",j,st1.l,st1.u);
   rtheta=rmax2(rtheta,rmaxabs(&st1)),
 }

/* higher order term */
 vsetzo(&v0t1);
 for(j=jmax+1;j<=n;j++)
   sscalar(rzero,rone,v0t1.p+j);
 sscalar(rzero,rone,&v0t1.h);
 dm(&v0t1,&v0t2);
 snorm(&v0t2,&st1);
 printf("higher order %27.16e %30.16e\n\n",st1.l,st1.u);
 rtheta=rmax2(rtheta,rmaxabs(&st1));

/* check if contraction */
 if(repsilon<rdcprod(rddiff(rone,rtheta),rbeta)){
   printf("**********************\n");
   printf("* contraction on ball *\n");
   printf("**********************\n");
 }
}

/* Include the subroutines */

/* initialisation routines */
#include "Section5.3"

/* the contraction M */
#include "Section5.4"

/* routines involving borel-functions */
#include "Section5.5"

/* vector operations */
#include "Section5.6"

/* routines acting on scalars */
#include "Section5.7"


/* 5.2. TYPES AND EXTERNAL VARIABLES */
/* ================================= */

#include <stdio.h>

typedef int logical;
typedef double reps;

typedef struct s{
   reps l;
   reps u;
 }
 scalar;

typedef struct v{
   scalar p[degree_plus_one];
   scalar g;
   scalar h;
 }
 vector;

typedef struct b{
   scalar p[degree_plus_one];
   scalar g;
   scalar h;
   reps e;
 }
```

```
 borel_function;

int n,n1;

reps
  rrho,r0save,r1save,r2save,r3save,
  rzero =0.0e+00,
  rone =1.0e+00,
  rtwo =2.0e+00,
  rthree=3.0e+00,
  rfour =4.0e+00,
  rhalf =0.5e+00,
  r3half=1.5e+00;

scalar
  sa,srho,sdnu,srarg,snormal,
  s2,s3,s4,si2,si3,si4,
  szero ={0.0e+00, 0.0e+00},
  sone ={1.0e+00, 1.0e+00},
  stwo ={2.0e+00, 2.0e+00},
  sthree={3.0e+00, 3.0e+00},
  sfour ={4.0e+00, 4.0e+00},
  shalf ={0.5e+00, 0.5e+00},
  s3half={1.5e+00, 1.5e+00};

vector
  ve,varg,vfact,vhhat,vpower[degree_plus_one];

borel_function bhto7;


/* 5.3. INITIALISATION ROUTINES */
/* ============================ */

/* read_approx_fixed_point     */
/* init_pack_and_unpack        */
/* init_multiplication_by      */
/* init_composition_with       */


read_approx_fixed_point(pv)
vector *pv;
{
  extern FILE
    *fopen();

  FILE *fp;

  int i;

  double d;

  fp=fopen("Section5.9","r");
  fscanf(fp,"%*86c");
  for(i=0;i<=n;i++){
    fscanf(fp,"%lf",&d);
    srconst(d,pv->p+i);
  }
  fclose(fp);

  sequal(&szero,&pv->g);
  sequal(&szero,&pv->h);
}

init_pack_and_unpack()
{
  extern scalar
    sa,s2,s3,s4,si2,si3,si4;

  extern vector
    ve;

/* make approximate eigenvector ve */
  vsetzo(&ve);
  srconst(rone,ve.p+1);
  srconst(.6e+00,ve.p+2);

/* choose coordinates */
  srconst(approx_linear_coeff,&sa);
  siconst(1,&s2);
  siconst(3,&s3);
  siconst(2,&s4);
  sinv(&s2,&si2);
  sinv(&s3,&si3);
  sinv(&s4,&si4);
}

init_multiplication_by(pv)
vector *pv;
{
  extern scalar
    srho;

  scalar s;

/* produce  v(z)=(1+z*rho/33)**(-1/2) */
  siconst(33,&s);
  squot(&srho,&s,&s);
  vsetms(pv,&s);
}

init_composition_with(pv)
vector *pv;
{
  extern reps
    r0save,r1save,r2save,r3save,
    rmaxabs(),rnorm();

  extern scalar
    srho,srarg;

  extern vector
    vpower[];

  int
    i,j,k,ii,kh,nh;

  reps
    r1,r2,r3;

  scalar
    st1,st2;

  vector
    vt1,vt2,vt3,vz,vs[6];

  srconst(rone/scaling_factor,&srarg);

/* produce the vector */
/* v(z)=((3/11+z*rho/22)/(1+z*rho/33))/(r*rho) */
  siconst(33,&st1);
  squot(&srho,&st1,&st1);
  vsetin(&vz,&st1);
  siconst(11,&st1);
  sprod(&st1,&srho,&st1);
  squot(&sthree,&st1,&st1);
  siconst(22,&st2);
  sinv(&st2,&st2);
  vsetli(&vt1,&st1,&st2);
  vmult(&vt1,&vz,&vz);
  sinv(&srarg,&st1);
  vsmult(&vz,&st1,&vz);
  vequal(&vz,pv);

/* save powers of vector v */
  vmult(&vz,&vz,&vt2);
  j=2;
  i=1;
  while(j<=n1){
    vequal(&vt2,&vs[i++]);
    vmult(&vt2,&vt2,&vt2);
    j=j+j;
  }
  ii=1;
  vequal(&vz,&vpower[ii++]);
  r1=rmax2(rone,rnorm(0,&vz));
  r2=rnorm(1,&vz);
  for(i=3;i<=n1;i+=2){
    k=(i-1)/2;
    j=1;
    vsetli(&vt2,&sone,&szero);
    while(k>0){
```

```
    kh=k/2;
    if(kh+kh!=k){
      vequal(&vs[j],&vt3);
      vmult(&vt2,&vt3,&vt2);
    }
    k=kh;
    ++j;
  }
  vequal(&vt2,&vpower[ii++]);
  r1=rmax2(r1,rnorm(0,&vt2));
  r2=rmax2(r2,rnorm(1,&vt2));
  if(ii<=n){
    vmult(&vt2,&vz,&vt3);
    vequal(&vt3,&vpower[ii++]);
    r1=rmax2(r1,rnorm(0,&vt3));
    r2=rmax2(r2,rnorm(1,&vt3));
  }
 }
 nh=n/2;
 if(nh+nh==n)
  r3=rnorm(0,&vt3);
 else
  r3=rnorm(0,&vt2);
 if(r3>rone)
  printf("error in vcomp, norm = %30.16e\n",r3);

/* save constants */
 r0save=rmaxabs(vz.p);
 r1save=r1;
 r2save=r2;
 r3save=r3;
}


/* 5.4. THE CONTRACTION M */
/* ====================== */

/* m       : the contraction                    */
/* dm      : the tangent map of m               */
/* r       : the RG transformation              */
/* dr      : the tangent map of r               */
/* unpack  : unpack operator used in m          */
/* dunpack : unpack operator used in dm         */
/* wzero   : part of unpack and dunpack         */
/* pack    : the left inverse of unpack         */
/* u_param : computes parameter for unpack      */
/* w_param : computes parameter for dunpack     */


m(pv0,pv0i)
vector *pv0,*pv0i;
{
 scalar snu;

 vector vt1;

 unpack(pv0,&snu,&vt1);
 r(&vt1,&vt1);
 pack(&snu,&vt1,pv0i);
}

dm(pv0,pv0i)
vector *pv0,*pv0i;
{
 scalar snu0;

 vector vt1;

 dunpack(pv0,&snu0,&vt1);
 dr(&vt1,&vt1);
 pack(&snu0,&vt1,pv0i);
}

r(pv,pvi)
vector *pv,*pvi;
{
 extern scalar
  snormal;

 extern vector
  vhhat;

 extern borel_function
  bhto7;

 scalar
  st1,st2;

 vector
  vt1,vt2;

 borel_function
  bt1,bt2,bt4,bt8;

/* take borel transform */
 borel(pv,&bt1);

/* take 8th power of bh */
 srconst(.02e+00,&st1);
 bmult(&bt1,&bt1,&st1,&bt2);
 srconst(.04e+00,&st1);
 bmult(&bt2,&bt2,&st1,&bt4);
 srconst(.04e+00,&st1);
 bmult(&bt4,&bt4,&st1,&bt8);

/* take inverse borel transform */
 iborel(&bt8,&vt1);

/* substitute varg in vh_hat */
 vargcmp(&vt1,&vt1);

/* multiply with vfact */
 vmult(&vt1,&vfact,&vt1);

/* normalize to one */
 sinv(vt1.p,&snormal);
 vsmult(&vt1,&snormal,&vhhat);
 sequal(&sone,vhhat.p);
 vequal(&vhhat,pvi);

/* save 7th power of bh */
 srconst(.03e+00,&st1);
 bmult(&bt4,&bt2,&st1,&bhto7);
 srconst(.03e+00,&st1);
 bmult(&bhto7,&bt1,&st1,&bhto7);
 siconst(8,&st1);
 bsmult(&bhto7,&st1,&bhto7);
}

dr(pv,pvi)
vector *pv,*pvi;
{
 extern scalar
  snormal;

 extern vector
  vhhat;

 extern borel_function
  bhto7;

 scalar st1;

 vector
  vt1,vt2;

 borel_function bt1;

/* take borel transform */
 borel(pv,&bt1);

/* multiply with 7th power of bh */
 srconst(.04e+00,&st1);
 bmult(&bt1,&bhto7,&st1,&bt1);

/* take inverse borel transform */
 iborel(&bt1,&vt1);

/* substitute varg in vh_hat */
 vargcmp(&vt1,&vt1);

/* multiply with vfact */
```

```
  vmult(&vt1,&vfact,&vt1);

/* renormalize */
  vsmult(&vhhat,vt1.p,&vt2);
  vminus(&vt2,&vt1);
  sequal(&szero,vt1.p);
  vsmult(&vt1,&snormal,&vt1);
  vequal(&vt1,pvi);
}

unpack(pv0,ps,pv)
vector *pv0,*pv;
scalar *ps;
{
  vector vt1;

  wzero(pv0,pv);
  sequal(&sone,pv->p);
  sequal(&sa,pv->p+1);
  u_param(pv,ps);
  vsmult(&ve,ps,&vt1);
  vadd(&vt1,pv);
}

dunpack(pv0,ps,pv)
vector *pv0,*pv;
scalar *ps;
{
  vector vt1;

  wzero(pv0,pv);
  w_param(pv,ps);
  vsmult(&ve,ps,&vt1);
  vadd(&vt1,pv);
}

wzero(pv0,pv)
vector *pv0;
vector *pv;
{
  int i;

  reps
    rl,ru;

  scalar st1;

  sequal(&szero,pv->p);
  sequal(&szero,pv->p+1);
  sprod(pv0->p+2,&si2,pv->p+2);
  sprod(pv0->p+3,&si3,pv->p+3);
  sprod(pv0->p+4,&si4,pv->p+4);

  for(i=5;i<=n;i++)
    sequal(pv0->p+i,pv->p+i);

  ru=rmax2(rmax3(si2.u,si3.u,si4.u),rone);
  rl=rmin2(rmin3(si2.l,si3.l,si4.l),rone);
  sscalar(rl,ru,&st1);

  sprod(&pv0->g,&st1,&pv->g);
  sequal(&pv0->h,&pv->h);
}

pack(ps,pv,pv0)
scalar *ps;
vector *pv,*pv0;
{
  int i;

  reps
    rl,ru;

  scalar st1;

  vector
    vt1,vt2;

  vequal(pv,&vt1);
  vsmult(&ve,ps,&vt2);
  vminus(&vt2,&vt1);

  sequal(&szero,pv0->p);
  sequal(&szero,pv0->p+1);
  sprod(vt1.p+2,&s2,pv0->p+2);
  sprod(vt1.p+3,&s3,pv0->p+3);
  sprod(vt1.p+4,&s4,pv0->p+4);

  for(i=5;i<=n;i++)
    sequal(vt1.p+i,pv0->p+i);

  ru=rmax2(rmax3(s2.u,s3.u,s4.u),rone);
  rl=rmin2(rmin3(s2.l,s3.l,s4.l),rone);
  sscalar(rl,ru,&st1);

  sprod(&vt1.g,&st1,&pv0->g);
  sequal(&vt1.h,&pv0->h);
}

u_param(pv,ps)
vector *pv;
scalar *ps;
{
  extern logical
    lsequal();

  extern reps
    rmaxabs();

  extern scalar
    sdnu;

  int i;

  reps
    rt1,rt2,rrad,rdnu;

  scalar
    srad,seps,st1,st2,st3;

  vector
    vt1,vt2,vdnu;

  sequal(&szero,ps);

/* calculate precision of center */
  vequal(pv,&vt1);
  r(&vt1,&vt2);
  rt1=rmaxabs(&vt1.g);
  rt2=rmaxabs(&vt2.g);
  sscalar(-rt1,rt1,&st1);
  sscalar(-rt2,rt2,&st2);
  sadd(vt1.p+1,&st1);
  sadd(vt2.p+1,&st2);
  sequal(&st2,&seps);
  sminus(&st1,&seps);

/* estimate derivative on neighborhood */
  rrad=5.e-3;
  sscalar(-rrad,rrad,&srad);

  for(i=1;i<=2;i++){
    sequal(&srad,&st1);
    sadd(ps,&st1);
    vequal(pv,&vt1);
    vsmult(&ve,&st1,&vt2);
    vadd(&vt2,&vt1);
    r(&vt1,&vt2);
    dr(&ve,&vdnu);
    rt2=rmaxabs(&vdnu.g);
    sscalar(-rt2,rt2,&sdnu);
    sadd(vdnu.p+1,&sdnu);
    sequal(&sdnu,&st2);
    sminus(&sone,&st2);
    squot(&seps,&st2,&st2);
    sneg(&st2,&st2);
    sinter(&srad,&st2,&st3);
    if(!lsequal(&st3,&st2)){
      printf("error in u_param, no contraction. step %d\n",i);
      sequal(&st1,ps);
      return;
    }
```

```
  sequal(&st2,&srad);
 }
 sadd(&srad,ps);
}

w_param(pv,ps)
vector *pv;
scalar *ps;
{
 extern reps
  rmaxabs();

 extern scalar
  sdnu;

 reps rt1;

 scalar st1;

 vector vt1;

 dr(pv,&vt1);

 rt1=rmaxabs(&vt1.g);
 sscalar(-rt1,rt1,&st1);
 sadd(vt1.p+1,&st1);
 sequal(&sone,ps);
 sminus(&sdnu,ps);
 squot(&st1,ps,ps);
}


/* 5.5. ROUTINES INVOLVING BOREL FUNCTIONS */
/* ======================================= */

/* bsetzo   : set borel function equal zero      */
/* bequal   : copy borel function                */
/* bsmult   : scalar times borel function        */
/* bmult    : product of two borel functions     */
/* sabound  : bound used in bmult                */
/* borel    : the borel transform                */
/* iborel   : the inverse borel transform        */


bsetzo(pb)
borel_function *pb;
{
 int i;

 for(i=0;i<=n;i++)
  sequal(&szero,pb->p+i);
 sequal(&szero,&pb->g);
 sequal(&szero,&pb->h);
 pb->e=rzero;
}

bequal(pb,pbres)
borel_function *pb,*pbres;
{
 int i;

 for(i=0;i<=n;i++)
  sequal(pb->p+i,pbres->p+i);
 sequal(&pb->g,&pbres->g);
 sequal(&pb->h,&pbres->h);
 pbres->e=pb->e;
}

bsmult(pb,ps,pbres)
borel_function *pb,*pbres;
scalar *ps;
{
 int i;

 scalar s;

 for(i=0;i<=n;i++)
  sprod(ps,pb->p+i,pbres->p+i);

 sabs(ps,&s);
 sprod(&pb->g,&s,&pbres->g);
 sprod(&pb->h,&s,&pbres->h);
 pbres->e=pb->e;
}

bmult(pb1,pb2,pseps,pbres)
borel_function *pb1,*pb2,*pbres;
scalar *pseps;
{
 extern logical
  lsequal();

 extern reps
  rmaxabs();

 int
  i,k;

 scalar
  st1,st2,s1,s2,
  sb1,sb2,sb1eps,sb2eps,
  sa1g,sa1h,sa2g,sa2h,
  sagres,sbres;

 borel_function
  bt1;

 sequal(&pb1->g,&sa1g);
 sequal(&pb1->h,&sa1h);
 sequal(&pb2->g,&sa2g);
 sequal(&pb2->h,&sa2h);
 srconst(pb1->e,&sb1);
 srconst(pb2->e,&sb2);

/* compute polynomial part */
 bsetzo(&bt1);
 for(k=0;k<=n;k++){
  sequal(&szero,&s1);
  for(i=0;i<=k;i++){
   sprod(pb1->p+i,pb2->p+k-i,&s2);
   sadd(&s2,&s1);
  }
  sequal(&s1,bt1.p+k);
 }

/* determine constant in exponential */
 sequal(pseps,&sb1eps);
 sadd(&sb1,&sb1eps);
 sequal(pseps,&sb2eps);
 sadd(&sb2,&sb2eps);
 sequal(pseps,&sbres);
 sadd(&sb1,&sbres);
 sadd(&sb2,&sbres);

/* bound error terms */
 sequal(&szero,&sagres);

 for(k=n1;k<=n+n;k++){
  sequal(&szero,&st1);
  for(i=k-n;i<=n;i++){
   sprod(pb1->p+i,pb2->p+k-i,&st2);
   sadd(&st2,&st1);
  }
  sabs(&st1,&st1);
  sabound(&sbres,k-1,&st2);
  sprod(&st1,&st2,&st1);
  sadd(&st1,&sagres);
 }

 for(i=0;i<=n;i++){
  sabs(pb1->p+i,&st1);
  sprod(&st1,&sa2g,&st1);
  sabound(&sb1eps,i,&st2);
  sprod(&st1,&st2,&st1);
  sadd(&st1,&sagres);
  sabs(pb1->p+i,&st1);
  sprod(&st1,&sa2h,&st1);
  sabound(&sb1eps,i+n,&st2);
  sprod(&st1,&st2,&st1);
  sadd(&st1,&sagres);
  sabs(pb2->p+i,&st1);
  sprod(&st1,&sa1g,&st1);
```

```
    sabound(&sb2eps,i,&st2);
    sprod(&st1,&st2,&st1);
    sadd(&st1,&sagres);
    sabs(pb2->p+i,&st1);
    sprod(&st1,&sa1h,&st1);
    sabound(&sb2eps,i+n,&st2);
    sprod(&st1,&st2,&st1);
    sadd(&st1,&sagres);
  }

  sabound(pseps,1,&st1);
  sprod(&sa1g,&sa2g,&st2);
  sprod(&st1,&st2,&st1);
  sadd(&st1,&sagres),

  if(!lsequal(&sa1h,&szero) &&
     !lsequal(&sa2h,&szero)){
    sabound(pseps,n1+n,&st1),
    sprod(&sa1h,&sa2h,&st2);
    sprod(&st1,&st2,&st1);
    sadd(&st1,&sagres);
  }

/* return result */
  sscalar(rzero,rmaxabs(&sagres),&bt1.g);
  srconst(rzero,&bt1.h);
  bt1.e=rmaxabs(&sbres);
  bequal(&bt1,pbres);
}

sabound(pseps,k,ps)
scalar *pseps,*ps;
int k;
{
  scalar
    st1,st2,sexp1;

  if(k==0)
    sequal(&sone,ps);
  else{
    srconst(2.718281828e+00,&sexp1);
    sprod(pseps,&sexp1,&st1);
    siconst(k,&st2);
    squot(&st2,&st1,&st1);
    spower(&st1,k,&st1);
    srconst(st1.u,ps);
  }
}

borel(pv,pb)
vector *pv;
borel_function *pb;
{
  extern reps
    rmaxabs();

  int i;

  scalar
    st1,st2;

  sequal(&sone,&st1);
  sequal(&shalf,&st2);
  for(i=0;i<=n;i++){
    squot(pv->p+i,&st1,pb->p+i),
    sprod(&st1,&st2,&st1);
    sadd(&sone,&st2);
  }
  squot(&pv->g,&shalf,&st2);
  sscalar(rzero,rmaxabs(&st2),&pb->g);
  squot(&pv->h,&st1,&st2);
  sscalar(rzero,rmaxabs(&st2),&pb->h);
  pb->e=rone,
}

iborel(pb,pv)
borel_function *pb;
vector *pv;
{
  extern reps
    rmaxabs(),

  int i;

  reps
    rfive=5 e+00;

  scalar
    st1,st2,st3,sag,sb,
    srhohat,sfact,stdiff;

  sequal(&sone,&st1);
  sequal(&shalf,&st2);
  for(i=0;i<=n,i++){
    sprod(pb->p+i,&st1,pv->p+i);
    sprod(&st1,&st2,&st1);
    sadd(&sone,&st2);
  }
  sequal(&st1,&sfact),

/* extract exponential growth */
/* and determine domains */
  srconst(pb->e,&sb);
  sprod(&sb,&srarg,&st1);

  if(st1 u>rone)
    printf("domains in iborel incompatible %80.16\n",st1.u);

  sprod(&s3half,&st1,&st1),
  sadd(&sone,&st1);
  squot(&st1,&sb,&st1);
  srconst(rtwo/rfive,&st2);
  sprod(&st1,&st2,&st1);
  srconst(st1.l,&srhohat);

/* adapt polynomial part to this domain */
  sequal(&sone,&st1);
  for(i=0;i<=n;i++){
    sprod(pv->p+i,&st1,pv->p+i);
    sprod(&st1,&srarg,&st1);
  }

/* calculate error terms on this domain */
  srconst(pb->g.u,&sag);
  sprod(&sag,&shalf,&st1),
  sprod(&sb,&srhohat,&st2);
  sequal(&sone,&stdiff);
  sminus(&st2,&stdiff);
  ssqrt(&stdiff,&stdiff);
  spower(&stdiff,3,&st2);
  squot(&srarg,&st2,&st2);
  sprod(&st1,&st2,&st1);
  squot(&srarg,&srhohat,&st2);
  sequal(&sone,&st3);
  sminus(&st2,&st3);
  squot(&st1,&st3,&st1);
  sscalar(rzero,rmaxabs(&st1),&pv->g);
  sequal(&szero,&pv->h);
}


/* 5.6. OPERATIONS ON VECTORS */
/* ========================== */

/* vsetzo   . set vector equal zero              */
/* snorm    . norm of vector                     */
/* rnorm    . bound partial norm of vector       */
/* vequal   . copies vector                      */
/* vadd       sum of vectors                     */
/* vminus   . difference of vectors              */
/* vsmult   . product of vector and scalar       */
/* vmult    . product of two vectors             */
/* vargcmp  : substitute varg in vector          */
/* vsetlz   . create vector v(z)=c+lz            */
/* vsetin   . create vector v(z)=1/(1+sz)        */
/* vsetms   . set vector v(z)=(1+sz)**(-1/2)     */


vsetso(pv)
vector *pv;
{
  int i;
```

```
  for(i=0;i<=n;i++)
    sequal(&szero,pv->p+i);
  sequal(&szero,&pv->g);
  sequal(&szero,&pv->h);
}

snorm(pv,psres)
vector *pv;
scalar *psres;
{
  int i;

  scalar
    s,st1;

  sequal(&szero,&s);
  for(i=0;i<=n,i++){
    sabs(pv->p+i,&st1);
    sadd(&st1,&s);
  }
  sadd(&pv->g,&s);
  sadd(&pv->h,&s);
  sequal(&s,psres);
}

reps rnorm(imin,pv)
int imin;
vector *pv;
{
  extern reps
    rmaxabs();

  int i;

  scalar
    s,st1;

  sequal(&szero,&s);
  for(i=imin;i<=n;i++){
    sabs(pv->p+i,&st1);
    sadd(&st1,&s);
  }
  sadd(&pv->g,&s);
  sadd(&pv->h,&s);
  return(rmaxabs(&s));
}

vequal(pv,pvres)
vector *pv,*pvres;
{
  int i;

  for(i=0;i<=n;i++)
    sequal(pv->p+i,pvres->p+i);
  sequal(&pv->g,&pvres->g);
  sequal(&pv->h,&pvres->h);
}

vadd(pv,pvres)
vector *pv,*pvres;
{
  int i;

  for(i=0;i<=n;i++)
    sadd(pv->p+i,pvres->p+i);
  sadd(&pv->g,&pvres->g);
  sadd(&pv->h,&pvres->h);
}

vminus(pv,pvres)
vector *pv,*pvres;
{
  extern reps
    rminabs(),rmaxabs();

  int i;

  scalar
    st1,st2;

  for(i=0;i<=n;i++)
    sminus(pv->p+i,pvres->p+i),

  sequal(&pv->g,&st1),
  sminus(&pvres->g,&st1);
  sequal(&pv->g,&st2);
  sadd(&pvres->g,&st2);
  sscalar(rminabs(&st1),rmaxabs(&st2),&pvres->g);

  sequal(&pv->h,&st1),
  sminus(&pvres->h,&st1),
  sequal(&pv->h,&st2);
  sadd(&pvres->h,&st2);
  sscalar(rminabs(&st1),rmaxabs(&st2),&pvres->h),
}

vsmult(pv,ps,pvres)
vector *pv,*pvres;
scalar *ps;
{
  int i;

  scalar s;

  for(i=0;i<=n;i++)
    sprod(pv->p+i,ps,pvres->p+i);

  sabs(ps,&s);
  sprod(&pv->g,&s,&pvres->g);
  sprod(&pv->h,&s,&pvres->h);
}

vmult(pva,pvb,pvres)
vector *pva,*pvb,*pvres;
{
  int
    i,k;

  scalar
    s1,s2,s3,st1,st2,
    seg,seh,sag,sah,sbg,sbh,
    sx[degree_plus_one];

  sequal(&pva->g,&sag);
  sequal(&pva->h,&sah);
  sequal(&pvb->g,&sbg);
  sequal(&pvb->h,&sbh);

/* calculate polynomial part */
  for(i=0;i<=n;i++)
    sequal(&szero,sx+i);

  for(k=0;k<=n;k++){
    sequal(&szero,&s1);
    for(i=0;i<=k;i++){
      sprod(pva->p+i,pvb->p+k-i,&s2);
      sadd(&s2,&s1);
    }
    sequal(&s1,sx+k);
  }

/* estimate higher order error */
  sequal(&szero,&seh),
  for(k=n1;k<=n+n;k++){
    sequal(&szero,&s1);
    for(i=k-n;i<=n;i++){
      sprod(pva->p+i,pvb->p+k-i,&s2);
      sadd(&s2,&s1);
    }
    sabs(&s1,&s1);
    sadd(&s1,&seh);
  }
  sequal(&szero,&s1);
  sequal(&szero,&s2),
  for(i=1;i<=n;i++){
    sabs(pva->p+n-i,&s3);
    sadd(&s3,&s1);
    sabs(pvb->p+n-i,&s3);
    sadd(&s3,&s2);
  }
  sabs(pva->p+n,&st1);
```

```
  sadd(&s1,&st1);
  sprod(&sbh,&st1,&st1);
  sabs(pvb->p+n,&st2);
  sadd(&s2,&st2);
  sprod(&sah,&st2,&st2);
  sadd(&st2,&st1),
  sprod(&sah,&sbh,&st2);
  sadd(&st2,&st1);
  sprod(&sah,&sbg,&st2);
  sadd(&st2,&st1),
  sprod(&sag,&sbh,&st2);
  sadd(&st2,&st1),
  sabs(pvb->p+n,&st2);
  sprod(&st2,&sag,&st2),
  sadd(&st2,&st1);
  sabs(pva->p+n,&st2);
  sprod(&st2,&sbg,&st2),
  sadd(&st2,&st1);
  sadd(&st1,&seh);

/* estimate general error */
  sprod(&s2,&sag,&st1),
  sprod(&s1,&sbg,&st2);
  sadd(&st2,&st1);
  sprod(&sag,&sbg,&st2),
  sadd(&st2,&st1);
  sequal(&st1,&seg);

/* return result */
  for(i=0;i<=n;i++)
    sequal(sx+i,pvres->p+i);
  sequal(&seg,&pvres->g);
  sequal(&seh,&pvres->h);
}

vargcmp(pv,pvres)
vector *pv,*pvres,
{
  extern reps
    rmaxabs();

  int i;

  reps rt1;

  scalar
    st1,st2;

  vector
    vt1,vt2;


/* calculate polymonial part */
  vsetzo(&vt1),
  for(i=1;i<=n;i++){
    vequal(&vpower[n1-i],&vt2);
    vsmult(&vt2,pv->p+n1-i,&vt2);
    vadd(&vt2,&vt1),
  }
  sadd(pv->p,vt1.p);

/* add error terms */
  srconst(r0save,&st1);
  sabs(&st1,&st1);
  spower(&st1,n1,&st1);
  srconst(r0save,&st2);
  sabs(&st2,&st2),
  sprod(&st2,&pv->g,&st2);
  sprod(&st1,&pv->h,&st1);
  sadd(&st2,&st1);
  rt1=rmaxabs(&st1);
  sscalar(-rt1,rt1,&st1);
  sadd(&st1,vt1.p);
  srconst(r3save,&st1);
  srconst(r1save,&st2),
  sprod(&st1,&st2,&st1);
  sprod(&st1,&pv->h,&st1);
  srconst(r2save,&st2);
  sprod(&st2,&pv->g,&st2),
  sadd(&st2,&st1),
  rt1=rmaxabs(&st1);
  sscalar(rzero,rt1,&st1);
  sadd(&st1,&vt1.g),

  vequal(&vt1,pvres);

}

vsetli(pv,psc,psl)
vector *pv,
scalar *psc,*psl;
{
  vsetzo(pv);
  sequal(psc,pv->p),
  sequal(psl,pv->p+1),
}

vsetin(pv,ps)
vector *pv;
scalar *ps;
{
  extern reps
    rmaxabs();

  int i;

  reps
    rt1=rmaxabs(ps),

  scalar
    st1,st2,st3;

  if(rt1>=rone)
    printf("error in vsetin, radius= %30.16e\n",rt1);
  sequal(&sone,pv->p);
  for(i=2;i<=n1;i++){
    sprod(pv->p+i-2,ps,&st1);
    sneg(&st1,pv->p+i-1);
  }
  sequal(&szero,&pv->g);
  sabs(ps,&st1);
  spower(&st1,n1,&st2);
  sequal(&sone,&st3);
  sminus(&st1,&st3);
  squot(&st2,&st3,&st1);
  sscalar(rzero,st1 u,&pv->h);
}

vsetms(pv,ps)
vector *pv;
scalar *ps;
{
  extern reps
    rmaxabs();

  int i;

  reps
    rt1=rmaxabs(ps),

  scalar
    st1,st2,si,sf;

  if(rt1>=rone)
    printf("error in vsetms, radius=  %30.16e\n",rt1),

  sequal(&sone,pv->p);
  sneg(&shalf,&sf);
  sprod(&sf,ps,&sf);
  for(i=2;i<=n1,i++){
    sequal(&sf,pv->p+i-1);
    siconst(i,&si);
    sequal(&shalf,&st1);
    sminus(&si,&st1);
    squot(&st1,&si,&st1);
    sprod(&sf,&st1,&sf),
    sprod(ps,&sf,&sf),
  }
  sequal(&szero,&pv->g);
  sequal(&sone,&st1);
  sabs(ps,&st2);
  sminus(&st2,&st1),
```

```
  sabs(&sf,&st2);
  squot(&st2,&st1,&st1);
  sscalar(rzero,st1.u,&pv->h);
}

/* 5.7. ROUTINES ACTING ON SCALARS */
/* ============================== */

/* lsequal  : compare two scalars              */
/* rmaxabs  : largest distance from zero       */
/* rminabs  : smallest distance from zero      */
/* siconst  : convert integer to scalar        */
/* srconst  : convert rep to scalar            */
/* sscalar  : interval with given boundary     */
/* sabs     : absolute value of scalar         */
/* sneg     : zero minus scalar                */
/* sinv     : one divided by scalar            */
/* sequal   : copies scalar                    */
/* sinter   : intersection of two scalars      */
/* sadd     : sum of two scalars               */
/* sminus   : difference of two scalars        */
/* sprod    : product of two scalars           */
/* squot    : quotient of two scalars          */
/* spower   : scalar to the power k            */
/* ssqrt    : square root of scalar            */

logical lsequal(ps1,ps2)
scalar *ps1,*ps2;
{
  return((ps1->l==ps2->l &&
          ps1->u==ps2->u)? 1:0);
}

reps rmaxabs(ps)
scalar *ps;
{
  return(rmax2(-ps->l,ps->u));
}

reps rminabs(ps)
scalar *ps;
{
  return(rmax3(rzero,ps->l,-ps->u));
}

siconst(i,ps)
int i;
scalar *ps;
{
  ps->l=(reps)i;
  ps->u=(reps)i;
}

srconst(r,ps)
reps r;
scalar *ps;
{
  ps->l=r;
  ps->u=r;
}

sscalar(rl,ru,ps)
reps rl,ru;
scalar *ps;
{
  ps->l=rl;
  ps->u=ru;
}

sabs(ps,psres)
scalar *ps,*psres;
{
  reps
    rl=ps->l,
    ru=ps->u;

  psres->l=rmax3(rzero,rl,-ru);
  psres->u=rmax2(-rl,ru);
}

sneg(ps,psres)
scalar *ps,*psres;
{
  reps
    r=-ps->u;

  psres->u=-ps->l;
  psres->l=r;
}

sinv(ps,psres)
scalar *ps,*psres;
{
  extern reps
    rup();

  if(ps->l <= rzero && rzero <= ps->u){
    printf("error in sinv %30.16e %30.16e\n",ps->l,ps->u);
    sequal(ps,psres);
  }
  else
    sscalar(rdquot(rone,ps->u),
            ruquot(rone,ps->l),psres);
}

sequal(ps,psres)
scalar *ps,*psres;
{
  psres->l=ps->l;
  psres->u=ps->u;
}

sinter(ps1,ps2,psres)
scalar *ps1,*ps2,*psres;
{
  reps
    rl=rmax2(ps1->l,ps2->l),
    ru=rmin2(ps1->u,ps2->u);

  if(rl>ru){
    printf("error: no intersection. s1,s2= \n");
    printf("%30.16e %30.16e %30.16e %30.16e \n"
          ,ps1->l,ps1->u,ps2->l,ps2->u);
    sequal(ps1,psres);
  }
  else
    sscalar(rl,ru,psres);
}

sadd(ps1,psres)
scalar *ps1,*psres;
{
  extern reps
    rup();

  reps
    r1l=ps1->l,
    r1u=ps1->u,
    r2l=psres->l,
    r2u=psres->u;

  if(r1l==rzero){
    if(r1u==rzero)
      ;
    else if(r2u==rzero)
      sscalar(r2l,r1u,psres);
    else
      sscalar(r2l,rusum(r1u,r2u),psres);
  }
  else if(r2l==rzero){
    if(r2u==rzero)
      sequal(ps1,psres);
    else if(r1u==rzero)
      sscalar(r1l,r2u,psres);
    else
      sscalar(r1l,rusum(r1u,r2u),psres);
  }
  else{
    if(r1u==rzero)
      sscalar(rdsum(r1l,r2l),r2u,psres);
```

```
    else if(r2u==rzero)
      sscalar(rdsum(r1l,r2l),r1u,psres);
    else
      sscalar(rdsum(r1l,r2l),
              rusum(r1u,r2u),psres);
  }
}

sminus(ps1,psres)
scalar *ps1,*psres;
{
  scalar s;

  sneg(ps1,&s);
  sadd(&s,psres);
}

sprod(ps1,ps2,psres)
scalar *ps1,*ps2,*psres;
{
  extern reps
    rup();

  reps
    r1u,rul,ruu,rll,
    r1l=ps1->l,
    r1u=ps1->u,
    r2l=ps2->l,
    r2u=ps2->u;

  if(r1u==rzero && r1l==r1u||
     r2u==rzero && r2l==r2u)
    sequal(&szero,psres);
  else if(r1l>=rzero){
    if(r2l>=rzero)
      sscalar(rdcprod(r1l,r2l),rucprod(r1u,r2u),psres);
    else if(r2u<rzero)
      sscalar(rdcprod(r1u,r2l),rucprod(r1l,r2u),psres);
    else
      sscalar(rdcprod(r1u,r2l),rucprod(r1u,r2u),psres);
  }
  else if(r1u<rzero){
    if(r2l>=rzero)
      sscalar(rdcprod(r1l,r2u),rucprod(r1u,r2l),psres);
    else if(r2u<rzero)
      sscalar(rdcprod(r1u,r2u),rucprod(r1l,r2l),psres);
    else
      sscalar(rdcprod(r1l,r2u),rucprod(r1l,r2l),psres);
  }
  else{
    if(r2l==rzero)
      sscalar(rdcprod(r1l,r2u),rucprod(r1u,r2u),psres);
    else if(r2u<rzero)
      sscalar(rdcprod(r1u,r2l),rucprod(r1l,r2l),psres);
    else{
      r1u=rdcprod(r1l,r2u);
      rul=rdcprod(r1u,r2l);
      ruu=rucprod(r1u,r2u);
      rll=rucprod(r1l,r2l);
      sscalar(rmin2(r1u,rul),rmax2(ruu,rll),psres);
    }
  }
}

squot(ps1,ps2,psres)
scalar *ps1,*ps2,*psres;
{
  scalar s,

  sinv(ps2,&s);
  sprod(ps1,&s,psres);
}

spower(ps,k,psres)
scalar *ps,*psres;
int k;
{
  extern reps
    rmaxabs();

  int
    m,mh;

  scalar
    s1,st1;

  if(k<0)
    printf("error in spower, k not positive\n");
  else if(rmaxabs(ps)==rzero){
    if(k==0)
      sequal(&sone,psres);
    else
      sequal(&szero,psres);
  }
  else{
    m=k;
    sequal(ps,&s1);
    sequal(&sone,&st1);
    while(m>0){
      mh=m/2;
      if(mh+mh!=m){
sprod(&st1,&s1,&st1);
--m;
      }
      else{
sprod(&s1,&s1,&s1);
m=mh;
      }
    }
    sequal(&st1,psres);
  }
}

ssqrt(ps,psres)
scalar *ps,*psres;
{
  extern double
    sqrt();

  extern reps
    rup();

  reps
    rl,ru;

  if(ps->l>=rzero){
    rl=rmax2(rzero,sqrt(ps->l));
    while(rucprod(rl,rl)>ps->l)
      rl=rdown(rl);
    ru=rmax2(rzero,sqrt(ps->u));
    while(rdcprod(ru,ru)<ps->u)
      ru=rup(ru);
    sscalar(rl,ru,psres);
  }
  else{
    printf("error in sqrt, argument negative %30.16d\n",ps->
    sequal(&szero,psres);
  }
}


/* 5.8. OPERATIONS INVOLVING ONLY REPS */
/* ================================== */

/* rup      : successor in the set of reps      */
/* rdown    . predecessor in the set of reps    */
/* rmax2    . maximum of two reps               */
/* rmax3    : maximum of three reps             */
/* rmin2    : minimum of two reps               */
/* rmin3    : minimum of three reps             */
/* rusum    . upwards rounded sum of reps       */
/* rdsum    . downwards rounded sum of reps     */
/* rddiff   : downwards rounded diff. of reps   */
/* rucprod  : upper bound on product of reps    */
/* rdcprod  . lower bound on product of reps    */
/* ruquot   . upper bound on quotient of reps   */
/* rdquot   . lower bound on quotient of reps   */

/* Remark.
   The type reps (=double) is in 64-bit IEEE format.
   The standard IEEE format ( see [24] ) is as follows.
   Sign bit, 11-bit binary exponent with bias
```

```
   of 1023, 52-bit mantissa ( with "hidden" most
   significant bit=1 ). The mantissa represents a
   number < 2 but not < 1. A zero exponent means a
   value of zero, and the exponent 11111111111 means
   "not a number". Bytes are in reversed order;
   the lowest addressed byte is the least
   significant mantissa byte. */

reps rup(r)
reps r;
{
  union convert{
    reps r;
    unsigned long int i[2];
  };

  union convert
    con;

  unsigned long int
    low,exp_and_high;

  con.r=r;
  low=con.i[0];
  exp_and_high=con.i[1];

  if(low==0){
    if(exp_and_high==0)
      exp_and_high=0X00100000;
    else if(r>rzero)
      low=1;
    else if(exp_and_high==0X80100000)
      exp_and_high=0;
    else{
      low=0XFFFFFFFF;
      --exp_and_high;
    }
  }
  else if(r>rzero){
    if(low==0XFFFFFFFF){
      if(exp_and_high==0X7FFFFFF){
        printf("overflow error in up %30.16e\n",r);
        return(r);
      }
      else{
        low=0;
        ++exp_and_high;
      }
    }
    else
      ++low;
  }
  else
    --low;

  con.i[0]=low;
  con.i[1]=exp_and_high;
  return(con.r);
}

/* Reps macros */

#define rdown(r) (-rup(-(r)))

#define rmax2(x,y) (((x)>(y))? (x).(y))

#define rmax3(x,y,z) (((x)>(y))? rmax2(x,z).rmax2(y,z))

#define rmin2(x,y) (((x)<(y))? (x):(y))

#define rmin3(x,y,z) (((x)<(y))? rmin2(x,z):rmin2(y,z))

#define rusum(x,y) rup((x)+(y))

#define rdsum(x,y) rdown((x)+(y))

#define rddiff(x,y) (-rup((y)-(x)))

#define first(x,y) ((x)==rzero||(y)==rone)?(x)

#define second(x,y) ((x)==rone||(y)==rzero)?(y)

#define rucprod(x,y) (first(x,y).(second(x,y)·rup((x)*(y))))

#define rdcprod(x,y) (first(x,y).(second(x,y):rdown((x)*(y))))

#define ruquot(x,y) (first(x,y).rup((x)/(y)))

#define rdquot(x,y) (first(x,y).rdown((x)/(y)))

! 5.9 COEFFICIENTS OF THE
!     APPROXIMATE FIXED POINT
! ======================

   0.0000000000000000e+000
   0.0000000000000000e+000
   5.0910744110871690e-001
   9.5003399093995690e-001
   3.3399471598550260e-001
   7.5228788724207240e-002
   2.9166025360328340e-002
   9.7809075474681730e-003
   2.8435242149942920e-003
   7.1603661837487370e-004
   1.5546221099374870e-004
   2.8821170499654220e-005
   4.4811455122526670e-006
   5.6476122738011440e-007
   5.3552629703583600e-008
   3.0116983547275270e-009
  -5.6475304807471550e-011
  -3.0876243410073330e-011
  -2.5344800033482360e-012
   4.5983279951728720e-014
   2.5517826728299680e-014
   1.2620701467321130e-015
  -1.5954295084724120e-016
  -1.9577090887265150e-017
   5.6384521325252770e-019
   1.6946227528291020e-019
  -5.8050871684659000e-021
  -1.5252782196365650e-021
   3.5918686798167610e-022
   8.9115968244695160e-023
  -4.8724320918072530e-024
  -5.1256140504614330e-024
  -5.5171555999740080e-025
   2.0355234891894370e-025
   5.5448722017939900e-026
  -6.2413537679217950e-027
  -3.6414430696016150e-027
   1.6372588275807340e-028
   2.1526142159118610e-028
  -6.1726379503092100e-030
  -1.2428705431372500e-029
   5.9617241343875310e-031
   6.9991573813170870e-031
  -6.9906713111864140e-032
  -3.6182464583042550e-032
   6.9244538361489700e-033
   1.4733779930696580e-033
  -5.5893664296982620e-034
  -2.0798556973059780e-035
   3.5409929658431840e-035
  -3.7338449518599320e-036
  -1.5159188525456280e-036
   4.6350294425125140e-037
   9.9086553208077940e-039
  -2.9079748380155910e-038
   4.9369413872234490e-039
```

5.10. OUTPUT OF THE PROGRAM

```
NORM OF DIFFERENCE :

epsilon          -1.8246895661559250e-032   9.7775556492167340e-012

BOUND ON DM :

column 2          3.7694842329324760e-001   7.3641322263893230e-001
column 3          5.3664316424268770e-001   5.8914742780986610e-001
column 4          4.9589276248309590e-001   5.3166452511130540e-001
column 5          4.0981788472088450e-001   4.5672910936799980e-001
column 6          1.3411764334767070e-001   1.7351670562741190e-001
higher order      0.0000000000000000e+000   5.0912756751644280e-001

**********************
* contraction on ball *
**********************
```

Acknowledgements. The authors would like to thank G. Gallavotti for helpful discussions and one of the authors (P.W.) would like to thank T. Spencer for his continued support and A. Sokal for his enthusiasm and for all the useful comments.

References

1. Eckmann, J.-P., Koch, H., Wittwer, P.: A computer-assisted proof of universality for area-preserving maps. Memoirs Am. Math. **47**, 289 (1984)
2. Eckmann, J.-P., Wittwer, P.: Computer methods and Borel summability applied to Feigenbaum's equation. Lect. Notes in Phys., Vol. **227**. Berlin, Heidelberg, New York, Tokyo: Springer 1985
3. Lanford, O.E. III.: A computer-assisted proof of Feigenbaum's conjectures. Bull. Am. Math. Soc. New Ser. **6**, 427 (1982)
4. Moore, R.E.: Interval analysis. Prentice-Hall, series in automatic computation (1966)
5. Moore, R.E.: Methods and applications of interval analysis. SIAM, Philadelphia (1979)
6. Sokal, A.: An improvement of Watson's theorem on Borel summability. J. Math. Phys. **21**, 261 (1980)
7. de la Llave, R., Lanford, O.E. III.: Work in progress
8. Eckmann, J.-P., Magnen, J., Sénéor, R.: Decay properties and Borel summability for the Schwinger functions in $P(\phi)_2$ theories. Commun. Math. Phys. **39**, 251 (1975)
9. Magnen, J., Sénéor, R.: Physe space cell expansion and Borel summability for the Euclidean ϕ_3^4 theory. Commun. Math. Phys. **56**, 237 (1977)
10. Collet, P., Eckmann, J.-P.: A renormalization group analysis of the hierarchical model in statistical physics. Lect. Notes in Phys. **74**, Berlin, Heidelberg, New York: Springer 1978
11. Gawedzki, K., Kupiainen, A.: Triviality of ϕ_4^4 and all that in a hierarchical model approximation. J. Stat. Phys. **29**, 683 (1982)
12. Gawedzki, K., Kupiainen, A.: Rigorous renormalization group and asymptotic freedom. In: Scaling and self-similarity in physics (renormalization in statistical mechanics and dynamics). Fröhlich, J. (ed.). Progress in Physics, Vol. 7, Boston, Basel, Stuttgart: Birkhäuser 1983
13. Bleher, P.M.: Usp. Mat. Nauk. **32**, 243 (1977)
14. Dyson, F.J.: Existence of a phase transition in a one-dimensional Ising ferromagnet. Commun. Math. Phys. **12**, 91–107 (1969)

15. Dyson, F.J.: Nonexistence of spontaneous magnetization in a one-dimensional Ising ferromagnet. Commun. Math. Phys. **12**, 212–215 (1969)
16. Bleher, P.M., Sinai, Ya.G.: Investigation of the critical point in models of the type of Dyson's hierarchical model. Commun. Math. Phys. **33**, 23 (1973)
17. Bleher, P.M., Sinai, Ya.G.: Critical indices for Dyson's asymptotically hierarchical models. Commun. Math. Phys. **45**, 347 (1975)
18. Wilson, K.G., Kogut, J.B.: The renormalization group and the ε expansion. Phys. Rep. **12**C, 75 (1974)
19. Wilson, K.: Renormalization group and critical phenomenon. II. Phase space cell analysis of critical behavior. Phys. Rev. B**4**, 3184 (1971)
20. Collet, P., Eckmann, J.-P., Hirsbrunner, B.: A numerical test of Borel summability in the ε-expansion of the hierarchical model. Phys. Lett. **71**B, 385 (1977)
21. Reed, M.C.: Functional analysis and probability theory. In: Velo, G., Wightman, A. (eds.). Constructive Quantum Field Theory. Lect. Notes in Phys., Vol. 25, Berlin, Heidelberg, New York: Springer 1976
22. Treves, F.: Topological vector spaces, distributions and kernels. New York, London: Academic Press 1967
23. Kernigham, B.W., Ritchie, D.M.: The *C* programming language. New York: Prentice-Hall, software series 1978
24. Stevenson, D.: IEEE Computer Society. A proposed standard for binary floating-point arithmetic. Draft. 8.0 of IEEE Task P754 (1981)
25. Benfatto, G., Cassandro, M., Gallavotti, G., Nicolò, F., Olivieri, E., Presutti, E., Scacciatelli, E.: Some probabilistic techniques in field theory. Commun. Math. Phys. **71**, 95–130 (1980)
26. Benfatto, G., Cassandro, M., Gallavotti, G., Nicolò, F., Olivieri, E., Presutti, E., Scacciatelli, E.: On the ultraviolet stability in the Euclidean scalar field theories. Commun. Math. Phys. **71**, 343 (1980)
27. Baker, G.: Analysis of hyperscaling in the Ising model by the high-temperature series method. Phys. Rev. B **15**, 1553–1559 (1977)
28. Baker, G., Kincaid, J.M.: The continuous spin Ising model, $g_0 : \phi_4^4$: field theory, and the renormalization group. J. Stat. Phys. **24**, 469–528 (1981)
29. Rosen, J.: The Ising model limit of ϕ^4 lattice fields. Proc. Am. Math. Soc. **66**, 114–118 (1977)
30. Caginalp, G.: The ϕ^4 lattice field theory as an asymptotic expansion about the Ising limit. Ann. Phys. **124**, 189–207 (1980)
31. Caginalp, G.: Thermodynamic properties of the ϕ^4 lattice field theory near the Ising limit. Ann. Phys. **126**, 500–511 (1980)
32. Constantinescu, F.: Expansion of the double-well model near the Ising model. Institut für angewandte Mathematik Johann-Wolfgang-Goethe Universität. Preprint (1980)
33. Constantinescu, F., Stroter, B.: The Ising limit of the double-well model. J. Math. Phys. **21**, 881–890 (1980)
34. LeGuillou, J., Zinn-Justin, J.: Critical exponents for the *n*-vector model in three dimensions from field theory. Phys. Rev. Lett. **39**, 95–98 (1977)
35. Nickel, B.: Hyperscaling and universality in three dimensions. Cargese lectures (1980)
36. Bender, C., Cooper, F., Guralnik, G., Roskies, R., Sharp, D.: Numerical computation of the renormalized effective potential in the strong coupling limit. Phys. Rev. D **23**, 2976 (1981)
37. See e.g. in: Handbook of Chemistry and Physics
38. Camp, W.J., Saul, D.M., Van Dyke, J.P., Wortis, M.: Series analysis of corrections to scaling for the spin-pair correlations of the spin-*s* Ising model: confluent singularities, universality and hyperscaling. Phys. Rev. B **14**, 3990–4001 (1976)
39. Bessis, D., Moussa, P., Turchetti, G.: Subdominant critical indices for the ferromagnetic susceptibility of the spin-$\frac{1}{2}$ Ising model. J. Phys. A **13**, 2763–2773 (1980)
40. Moussa, P.: Subdominant critical singularities in the *bcc* Ising model. J. Stat. Phys. **27**, 711–719 (1982)
41. Chen, J.-H., Fisher, M.E.: Unbiased estimation of corrections to scaling by partial differential approximants. Phys. Rev. Lett. **48**, 630–634 (1982)

42. Adler, J., Moshe, M., Privman, V.: Unbiased map of the temperature plane and its consequences. Phys. Rev. B **26**, 3958–3959 (1982)
43. Ferer, M., Velgakis, M.J.: Hyperscaling in the three-dimensional Ising model. Phys. Rev. B **27**, 2839–2854 (1983)
44. Gammel, J.L., Nuttall, J., Power, D.C.: To appear
45. George, M.J., Rehr, J.J.: Two-series approach to partial differential approximants: Three-dimensional Ising models. Phys. Rev. Lett. **53**, 2063–2066 (1984)
46. Chen, J.-H., Fisher, M.E.: To appear in J. Phys.
47. Guttmann, T.: In preparation

Communicated by J.-P. Eckmann

Received April 4, 1986