# USING REDUCE FOR REPLICA CALCULATIONS

**Paul Lukowicz**
Institut für Programmstrukturen
und Datenorganisation
Universität Karlsruhe
D-7500 Karlsruhe 1
Germany
lukowicz@ira.uka.de

**Klaus-Robert Müller**
GMD FIRST
Rudower Chaussee 5
D-1199 Berlin
Germany
klaus@first.gmd.de

**Werner M. Seiler**
Institut für Algorithmen und
Kognitive Systeme
Universität Karlsruhe
D-7500 Karlsruhe 1
Germany
kg04@dkauni2.bitnet

**Abstract** The REDUCE package REPPACK for performing replica calculations in the context of neural networks is presented. REPPACK provides the user with several functions needed to calculate partition functions and saddle point equations interactively. The application of REPPACK is demonstrated for the toy example of finitely many patterns ($p/N \to 0$) for the Hopfield model. The Gardner calculation with REPPACK is briefly outlined.

## 1. INTRODUCTION

Symmetric Hopfield networks [8] are employed as associative memories. They can store a set of patterns and perform fault-tolerant retrieval even from distorted input. The properties of Hopfield networks have been examined through large scale simulations determining important parameters like storage capacity and basins of attraction [4, 7, 10, 5, 18].

Amit et al. have found a way to calculate the storage capacity analytically for hebbian couplings [1] using statistical mechanics and the replica trick [15]. They take the spins $s_i$ as dynamical variables to compute the free energy. Gardner has calculated the storage capacity for a fully connected network, where the patterns are embedded with a certain embedding strength [6]. In her analysis which also uses the replica trick the couplings $w_{ij}$ are the dynamical variables.

Both of these so called replica calculations can be done for different pattern distributions [6, 9], for different types of models, e.g. $Q$-state, analog or diluted models [17, 11, 2, 19] or for generalisation problems [16]. They imply technical difficulties in the analysis and numerical problems in the evaluation of the saddle-point equations.

Of course we can not overcome the numerical difficulties with a computer algebra system. Our goal is to simplify the calculations leading to the saddle-point equations. They consist in every specific application of similar operations. REPPACK provides the user with functions supporting the computation of the partition function and the saddle-point equations in a replica (symmetric) calculation. In order to maintain a certain universality in our ansatz, the user has to do his calculations interactively. Thus it should be possible to perform different – even non-standard – calculations semi-automatically, i.e. the user expresses the individual steps by calls of procedures provided by REPPACK. REPPACK itself is a package developed in the computer algebra system REDUCE [14].

The next section presents the functionality of REPPACK; in the third section we demonstrate the application of our package to the toy example of $p/N \to 0$ and look at the standard Gardner calculation. Finally, some conclusions are given.

## 2. REPPACK

Replica calculations contain several typical operations. In REPPACK we have implemented general types of algebraic manipulations (cf. table and see also [12, 13]). The package is written in the symbolic mode of REDUCE. All procedures are nevertheless accessible from the algebraic mode. It supports the handling of vectors, matrices and tensors with variable (possibly infinite) dimensions. Elements of those structures can be accessed and modified by specifying an access pattern (e.g. diagonal elements or elements belonging to a certain block) and complex operations like determinant or inversion, rules can be specified.
The package allows flexible, easy-to-use access to parts of complicated expressions and simple alteration of their representation. This is especially important since a human user can immediately tell e.g. over which terms an averaging should take place or whether some terms in the exponentials factorize over a certain index. REPPACK solves such tasks according to certain matching or consistency rules suitable for replica calculations or rules explicitely given by the user.
REPPACK can perform Gauß linearisation and provides procedures to transform $\theta$ and $\delta$ functions into their integral representations. It contains a Gauß integrator which includes an automatic consistency check. For a given set of differentation variables, saddle-point equations are computed automatically. To enhance the readability of the results, we adapted the REDUCE-TEX-interface to our needs. The saddle-point equations can be written directly into a FORTRAN file through the FORTRAN interface of REDUCE. Then they can as usually be solved numerically.

| | |
|---|---|
| nvec, nmat, nstruct | definition of a vector, matrix or tensor with variable dimension sizes |
| contr | reduces the dimensionality of an object (e.g. contracts a matrix to a vector) |
| nprod, nsum | new operators for sums and products with variable bounds |
| dint | new operator for definite, multidimensional integration with variable bounds |
| sav, strace | new operators for the average and the trace over system configurations |
| grpxxx, grpx | selects all factors and operators from a specified operator. xxx can be any REPPACK operator name |
| to!*xxx, tox | changes an exponential function over sums to a product of exponential functions and vice versa. xx can be nsum, nprod, sum, prod. |
| op!*, op!*part | selects the specified operator or its argument in a complex expression. op!*, op!*part can be used both on the left and right side of an assigmnent |
| gaussaux | introduces auxiliary variables using the gauss trick |
| thetaauxaux, deltaaux, oneaux | introduces auxiliary variables using the integral representation of $\theta$, $\delta$ or unity |
| saddle!*eqs | computes the saddle point equations |
| gauss!*int | performs a gaussian integration (multidimensional) over a specified variable |

Table 1: REPPACK contains about 50 different functions and operators, some of them are described in this table.

## 3. CALCULATIONS WITH REPPACK

### 3.1 A Toy Example

In this section we demonstrate the use of REPPACK for a toy example, namely the $p/N \to 0$

limit of the Hopfield model [1, 7]. In this limit, a Hopfield network is trained for $p$ patterns $\sigma_i^\mu$ ($\mu = 1 \ldots p,\ i = \ldots N$) with the Hebb rule

$$w_{ij} = \frac{1}{N} \sum_\mu \sigma_i^\mu \sigma_j^\mu . \tag{1}$$

We look at the system in the thermodynamic limit $N \to \infty$, where the number of patterns will be kept fixed such that $\alpha = p/N$ becomes arbitrarily small. To evaluate the partition function $Z$, we start with some preliminary definitions[1]:

```
H  := NSUM(i,1,NN,beta*sigma(mu,i)*s(i));
```

$$h := \sum_{i=1}^{N} (s_i \cdot \sigma_{\mu,i} \cdot \beta) \tag{2}$$

and define the partition function as

```
Z  := EXP(-1/(2*NN*beta)*NSUM(mu,1,p,H**2));
```

$$z := e^{-\left( (\frac{1}{2}) \cdot \sum_{\mu=1}^{p} \left( \sum_{i=1}^{N} (s_i \cdot \sigma_{\mu,i} \cdot \beta)^2 \right) \cdot N^{-1} \cdot \beta^{-1} \right)} \tag{3}$$

where we omitted the diagonal terms $\exp(-\frac{1}{2}\beta p)$ that do not contribute to the saddle-point equations. Note that in the TEX representation of REDUCE capital letters become non-capital ones, if capital letters are necessary, we have to write e.g. $NN$ in the REDUCE code to obtain $N$ in the TEX representation. In order to average over the states $s_i$, we introduce the auxiliary variables $m_\mu$ linearising the quadratic term in $s_i$ via a Gauß linearisation.

```
Z1 := GAUSSAUX(Z,H,m(mu));
```

$$z_1 := \prod_{\mu=1}^{p} \left( \sqrt{\beta} \cdot \pi^{-(\frac{1}{2})} \cdot \sqrt{N} \cdot \int_{-\infty}^{\infty} \left( e^{-\left(\frac{1}{2} m_\mu^2 \cdot N \cdot \beta\right) + m_\mu \cdot \sum_{i=1}^{N} (s_i \cdot \sigma_{\mu,i} \cdot \beta)} \right) \, dm_\mu \right) \tag{4}$$

The next line is more tricky

```
Z3 := op!*(dint,m(mu),1,Z1) := fixrules(grpstrace(s,{-1,1},
          strace(s,{-1,1},op!*(dint,m(mu),1,to!*nprod(Z1,i))),{}) where sigtr);
```

We will explain it step by step. First, in the innermost command *strace*, we average over all $s_i \in \{\pm 1\}$ under the integral over $m_\mu$. The command *grpstrace* tells REDUCE to perform this trace operation only where it is necessary, i.e. over $\exp(m_\mu \sum_i s_i \cdot \sigma_{\mu,i} \cdot \beta)$. Again we want to emphasize that a human being can easily see which term should be averaged over, but this is highly non-trivial for a computer. The expression *sigtr* makes the system use the exp ln 2cosh representation of 2cosh[2] in its output, and the *fixrules* command forces REDUCE to use exp ln not only in its output but also in its internal representation. $op! * (dint, m(mu), 1, Z1) := \ldots$ tells REDUCE to substitute the result of the trace operation for the integral.

$$z_3 := \prod_{\mu=1}^{p} \left( \sqrt{\beta} \cdot \pi^{-(\frac{1}{2})} \cdot \sqrt{N} \cdot \int_{-\infty}^{\infty} \left( e^{-\left(\frac{1}{2} m_\mu^2 \cdot N \cdot \beta\right)} \cdot \prod_{i=1}^{N} \left( e^{ln(2cosh(m_\mu \cdot \sigma_{\mu,i} \cdot \beta))} \right) \right) \, dm_\mu \right) \tag{5}$$

We finally arrive at the point, where we can differentiate with respect to $m_\mu$

```
saddle!*eqs(op!*part(expt,e,1,to!*nsum(Z3,i))/-(N*beta),{m(mu)});
```

and get the saddle-point equations

[1]Note, that multiple indices are separated by commata in REDUCE e.g. $\sigma_{\mu,i} \sim \sigma_i^\mu$.

[2]This is of course contraproductive for a computer algebra system, since it usually tends to simplify formulae instead of making them more complicated, whereas for humans the exp ln representation is just a matter of convenience.

$$\left\{-\left(\sum_{i=1}^{N}\left(tanh\left(m_{\mu}\cdot\sigma_{\mu,i}\cdot\beta\right)\cdot\sigma_{\mu,i}\right)\cdot N^{-1}\right)+m_{\mu}=0\right\} \tag{6}$$

This example should have made clear that even such a simple calculation can pose unsolvable problems to a computer, if the matching rules and the routines for accessing terms and altering representations are not worked out properly.

### 3.2 The Gardner Calculation

To show that the application of our package is not restricted to toy problems, we will briefly outline a Gardner type calculation in REPPACK. Technical details would go beyond the scope of this paper and can be found elsewhere [12, 13]. The full programm code we used is shown in the appendix. We only show some typical steps. As in the other example the REPPACK operators have to be defined first: the embedding strength $\gamma_i^{\mu}$ measures how good the patterns are learned

```
gamma(a,mu,i) := nsum(j,1,NN,sigma(mu,i)*sigma(mu,j)*omega(a,i,j))/NN**(1/2);
```

$$\gamma_{a,\mu,i}:=\sum_{j=1}^{N}\left(\sigma_{\mu,i}\cdot\sigma_{\mu,j}\cdot\omega_{a,i,j}\right)\cdot N^{-1/2}, \tag{7}$$

and the replicated fractional phase space volume $\Omega_i=\langle V^n\rangle$ in the deterministic limit $T\to 0$

```
Om(i):= sav(sigma,{-1,1},eta*nprod(a,1,n,dint(-inf,inf,rho(a)*
nprod(mu,1,p,theta(gamma(a,mu,i)-k)),nprod(j,1,NN,omega(a,i,j)))));
```

$$\Omega_i:=\left\langle\left(\prod_{a=1}^{n}\int_{-\infty}^{\infty}\left(\rho_a\cdot\prod_{\mu=1}^{p}\theta\left(\sum_{j=1}^{N}\left(\sigma_{\mu,i}\cdot\sigma_{\mu,j}\cdot\omega_{a,i,j}\right)\cdot N^{-1/2}-k\right)\right)\ d\prod_{j=1}^{N}\omega_{a,i,j}\cdot\eta\right)\right\rangle \tag{8}$$

$\rho^a$ is the integration measure for the spherical model, $\eta$ is the normalisation constant. In the TEX notation of REPPACK $d\prod_{j=1}^{N}\omega_{a,i,j}$ stands for $\prod_{j=1}^{N}d\omega_{a,i,j}$. Multiple indices are separated by commata, e.g. $\sigma_{\mu,i}\sim\sigma_i^{\mu}$ and $k$ is the embedding stability of the patterns usually denoted by $\kappa$. Now the $\theta$ or step function is replaced by its integral representation using the *thetaaux* function and the averaging is moved inside the integral (several lines of code in the appendix).

$$\begin{aligned}\Omega_i\ &:=\ \int_{-\infty}^{\infty}\int_{k}^{\infty}\int_{-\infty}^{\infty}\left(\prod_{j=1}^{N}\prod_{\mu=1}^{p}\left\langle\left(e^{-\left(\sigma_{\mu,i}\cdot\sigma_{\mu,j}\cdot\sum_{a=1}^{n}(x_{a,\mu}\cdot\omega_{a,i,j})\cdot i\cdot N^{-1/2}\right)}\right)\right\rangle\cdot\prod_{\mu=1}^{p}\left(e^{\sum_{a=1}^{n}(x_{a,\mu}\cdot\lambda_{a,\mu})\cdot i}\right)\right)\\ &\quad\cdot d\prod_{\mu=1}^{p}\prod_{a=1}^{n}x_{a,\mu}\cdot\ d\prod_{\mu=1}^{p}\prod_{a=1}^{n}\frac{\lambda_{a,\mu}}{2\pi}\cdot\prod_{a=1}^{n}\rho_a\ d\prod_{j=1}^{N}\prod_{a=1}^{n}\omega_{a,i,j}\cdot\eta\end{aligned} \tag{9}$$

After averaging and rearranging terms we consider $G_2(E_a,F_{ab})$ similar to Gardner [6].

$$g_2:=\ln\left(\int_{-\infty}^{\infty}\left(e^{\sum_{a=1}^{n}(-(i\cdot E))+\sum_{a=1}^{n}\sum_{b=1}^{n}\left(\left(\frac{1}{2}\right)\cdot\omega_a\cdot\omega_b\cdot\delta_{a,b}\cdot i\cdot F+\omega_a\cdot\omega_b\cdot\delta_{a,b}\cdot i\cdot E-\left(\frac{1}{2}\right)\cdot\omega_a\cdot\omega_b\cdot i\cdot F\right)}\right)\ d\prod_{a=1}^{n}\omega_a\right) \tag{10}$$

For a replica symmetric ansatz we perform a Gauß integration

```
on mcd; on div; h5 := gauss!*int(g2(EE,FF),ll);
```

$$h_5:=\ln\left(\frac{\pi^{\frac{n}{2}}\cdot e^{\sum_{a=1}^{n}(-(i\cdot E))}\cdot\sqrt{F+2\cdot E}\cdot 2^{\frac{n}{2}}}{\sqrt{-\left((-(i\cdot F)-2\cdot i\cdot E)^n\cdot n\cdot F\right)+(-(i\cdot F)-2\cdot i\cdot E)^n\cdot F+2\cdot(-(i\cdot F)-2\cdot i\cdot E)^n\cdot E}}\right)$$

and obtain the saddle-point equations for $F$ and $E$ in the limit $n\to 0$ and $q\to 1$,

```
sa:=saddle!*eqs(fix(h5  where !*dosum)+N*(N-1)*FF*q(a,b),{FF,EE})$
```

which completes our second example.

## 4. CONCLUSIONS

Our goal was to provide with REPPACK a modular package which is able to grow according to the needs of users. After some further testing on non-trivial problems, it will become generally available. An important advantage of using a computer algebra system is much higher probability for correct results. Many of the operations used in the replica trick are highly error-prone in hand calculations. The reported results may seem somewhat clumsy to readers not acquainted with computer algebra, but we hope that REPPACK will enable non-specialists to try methods of statistical mechanics for their problems and specialists familiar with the system to speed up their calculation considerably. Our future interest will be dedicated to a further simplification of the calculation and the consideration of replica symmetry breaking schemes within REPPACK.

## APPENDIX

```
NMAT    sigma(p,N), lam(NN,p), x(NN,p), q(NN,NN),FF(a,b)$
NSTRUCT omega({NN,N,NN}),gamma({N,p,NN})$
NVEC s(N), rho(NN), Om(NN),EE(NN)$

gamma(a,mu,i) := nsum(j,1,NN,sigma(mu,i)*sigma(mu,j)*omega(a,i,j))/NN**(1/2);
Om(i):= sav(sigma,{-1,1},eta*nprod(a,1,n,dint(-inf,inf,rho(a)*nprod(mu,1,p,
                 theta(gamma(a,mu,i)-k)),nprod(j,1,NN,omega(a,i,j))))));
Om(i):=(op!*part(nprod,mu,1,Om(i)):=
                thetaaux(op!*part(nprod,mu,1,Om(i)),lam(a,mu),x(a,mu)));
Om(i) := grpx(sv,{a,sigma},tox(s,grpx(pp,{mu,a},tox(p,Om(i),j)),a));
Om(i) :=(op!*(sav,sigma(j,mu),1,Om(i)):=e**(-1/2*nsum(a,1,N,nsum(b,1,N,
       x(a,mu)*omega(a,i,j)*x(b,mu)*omega(b,i,j)/NN))));
Om(i) := tox(p,fix(grpx(s,j,tox(ss,Om(i),{mu,j})) where insq),mu);
Om(i) := grpx(ii,{lam(a,mu),x(a,mu)},Om(i));
HGG := op!*(nprod,mu,1,Om(i));
Om(i) := tosum((op!*(nprod,mu,1,Om(i)) := exp(NN*alpha*gg(q(a,b)))));
rho(a) := deltaaux(delta(nsum(j,1,NN,omega(a,i,j)**2-NN)),EE(a));
H0:=nprod(a,1,N,nprod(b,1,a-1,
    oneaux((1/NN*nsum(j,1,NN,omega(a,i,j)*omega(b,i,j))),q(a,b),FF(a,b)*NN)));
Om(i) := grpx(ppi,{b,a,omega(a,i,j)},
              (op!*(nprod,a,1,Om(i)) := op!*(nprod,a,1,Om(i))*H0));
HG20 := op!*(dint,nprod(j,1,NN,nprod(a,1,N,omega(a,i,j))),1,Om(i))$
hg2  := (op!*part(nsum,j,1,hg20) := omega(a,i,j)**2-1);
Om(i):=(op!*(dint,nprod(j,1,NN,nprod(A,1,N,omega(a,i,j))),
              1,Om(i)):=e**(g2(EE,FF)));
h3 := grpx(i,{omega(a,i,j)},tox(ss,grpx(pp,{b,a},tox(p,hg2,j)),{b,a}));
h4 := fix(contr({EE,FF,FF,omega,omega},h3,{1,2,1,3,2}) where {!*npf,!*mrs});
off mcd; g2(EE,FF) := fix(ln(stm(a,b,omega(a),h4)) where !*lnr);
on mcd; on div; h5 := gauss!*int(g2(EE,FF),ll);
sa:=saddle!*eqs(fix(h5  where !*dosum)+N*(N-1)*FF*q(a,b),{FF,EE})$
s1 := first(sa);s2 := second(sa);
off mcd; g1(x,lam):= fix(ln(stm(a,b,x(a),contr({q,q,x,lam},hgg,{2,2,2,2})))
                        where {!*npf,!*lnr});
on mcd; h8 :=gauss!*int(g1(x,lam),ppq)$
h9 := fix(h8 where n*q = 0);
h10 := mts(a,b,lam(a),si(fix(h8 where n*q = 0),{2*n->0,4*n->0})); off mcd;
```

```
h11 :=tox(p,gaussaux(h10,nsum(a,1,n,lam(a)),(-q**(1/2)/(1-q)),z),a);
on mcd; on div; h11; h12 := fix(contr(lam,grpx(i,{lam(a)},h11),1) where !*npf);
```

## References


[1] Amit, D.J., Gutfreund H., Sompolinsky, H., Phys. Rev. Lett. 55, 1530 (1985), Phys. Rev. A32,1007, (1985), Ann. Phys., NY 173, 30 (1987)

[2] Bouten, M., Engel, A., Komoda, A., Serneels, R., J. Phys. A:Math. Gen. 23, 4643 (1990)

[3] Buhmann, J., Divko, R., Schulten, K., Phys. Rev. A39, 2689 (1989)

[4] Forrest, B.M., J. Phys. A:Math. Gen. 21, 245 (1988)

[5] Forrest, B.M., Wallace, D.J., in: Models of Neural Networks, ed. Domany, E., van Hemmen, J.L. and Schulten K., Springer Heidelberg, 121 (1991)

[6] Gardner, E., J. Phys. A:Math. Gen. 21, 257 (1988)

[7] Hertz, J., Krogh, A., Palmer, R.G., Introduction to the Theory of Neural Computation, Addison-Wesley Redwood City (1991)

[8] Hopfield, J.J., Proc. Natl. Acad. Sci. USA, 79, 2554 (1982)

[9] Horner, H., Z. Phys. B 75, 133 (1989)

[10] Kohring, G.A., J. Stat. Phys. 59, 1077 (1990)

[11] Kühn, R., Bös, S., van Hemmen, J.L., Phys. Rev. A 43, 2084 (1991)

[12] Lukowicz, P., Anwendung von Computeralgebra auf Probleme Statistischer Physik neuronaler Netz, diploma thesis (in german), Universität Karlsruhe, Fakultät f. Physik (1993)

[13] Lukowicz, P., Müller, K.-R., Seiler, W.M., Application of Computeralgebra for Replica Calculations in Statistical Physics of Neural Networks, in preparation

[14] MacCallum, M.A.H., Wright, F.J., Algebraic Computing with REDUCE, Clarendon Press Oxford (1991)

[15] Mézard, M., Parisi, G., Virasoro, M.A., Spin Glass Theory and Beyond, World Scientific, Singapore, (1987)

[16] Opper, M., Kinzel, W., Kleinz, J., Nehl, R., J.Phys. A:Math.Gen. 23, L581 (1990)

[17] Rieger, H., J. Phys. A:Math. Gen. 23 L1273 (1990)

[18] Stiefvater, T., Müller, K.-R., J.Phys. A:Math.Gen. 25, 5919 (1992)

[19] van Hemmen, J.L., Kühn, R., in Models of Neural Networks, ed. Domany, E., van Hemmen, J.L. und Schulten K. (ed.), Springer Heidelberg, 1 (1991)