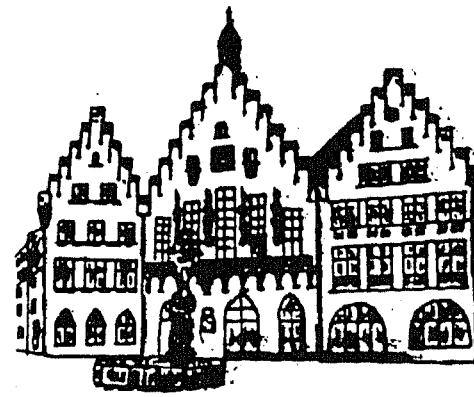
FACHBEREICH INFORMATIK

Universität Frankfurt

PERFORMANCE AND STORAGE REQUIREMENTS OF TOPOLOGY-CONSERVING MAPS FOR ROBOT MANIPULATOR CONTROL

DR. R. BRAUSE

INTERNER BERICHT 5/89



Performance and Storage Requirements of Topology-conserving Maps for Robot Manipulator Control

Dr. R. Brause
J.W.Goethe University, FB Informatik VSFT,
Postbox 111932, D - 6000 Frankfurt 11,
West-Germany

Abstract

A new programming paradigm for the control of a robot manipulator by *learning* the mapping between the Cartesian space and the joint space (inverse Kinematic) is discussed. It is based on a Neural Network model of optimal mapping between two high-dimensional spaces by Kohonen.
This paper describes the approach and presents the optimal mapping, based on the principle of maximal information gain. It is shown that Kohonens mapping in the 2-dimensional case is optimal in this sense.
Furthermore, the principal control error made by the learned mapping is evaluated for the example of the commonly used PUMA robot, the trade-off between storage resources and positional error is discussed and an optimal position encoding resolution is proposed.

1) Introduction

In the recent developements of real-time, parallel processing networks new ideas evolve by the introduction of the fine-grained parallelism of Neural Networks. The discovery of highly parallel and fault-tolerant models of the 30 year old brain research for computer science, combined with the new ULSI and wafer-scale integration possibilities of today's chip technology brings the human sensory and motor performance within the reach of artificial implementation.
This paper deals with one of these models, the so-called topology conserving maps. It is well known in neurophysiology that in the human brain there exists mappings between external sensory and effector signals and parts of the brain (*somatosensory mappings*). In figure 1a the mapping of muscles of the body to a part of the brain, called *Gyrus präcentralis*, is shown.
This paper discusses how a topology-conserving mapping can be used to learn the control of a robot manipulator. Contrary to the conventional, analytical solutions learning the manipulator positioning has some major advantages:

a) Since the geometry of the manipulator arm is not explicitely represented but implicitely learned, the control can easily be adapted to tolerate manipulator fabrication variations and worn-out joints without special reprogramming.

b) By this method it becomes now possible to control manipulators which have many joints (>3) with not simply (orthogonally or parallel) oriented rotation axes which is analytically very hard or impossible to treat. The necessary restrictions for the degrees of freedom can be easily incorperated in the learning rule without special overhead.

c) The calculation of the necessary joint angles for a certain desired cartesian position is done very fast, even in the case when the manipulator joint rotation axes are not oriented orthogonal or parallel.

d) The learned positioning allows features such as coarse positioning resolution in rare-used regions and fine resolution at often used locations as pick-ups etc., including obstacle avoiding. An example is given in figure 1b.

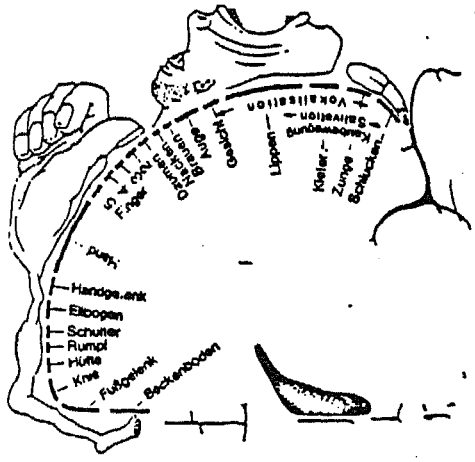


Fig.1a A somatotopic mapping (from [SCH])

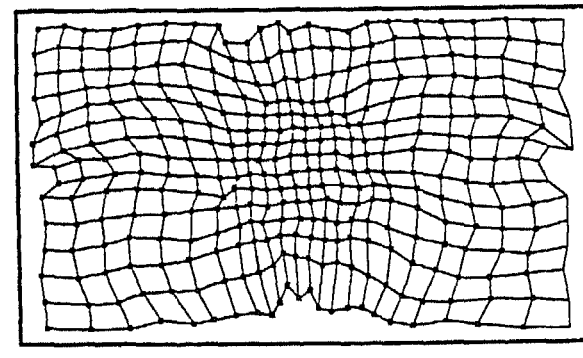


Fig.1b Dynamic pos. resolution [RITT3]

Here we see that the positioning resolution, indicated by a resolution grid, is automatically finer in the middle of the working area where the positioning more often took place (see section 3).

2.) Robot movement and the problem of inverse kinematics

In the standard control technique of robot manipulators the control of the joints is done by microprocessors and their associated servo power amplifiers. Each joint has also sensors (e.g. position encoding by optical encoders) which are used for the calculation of the correct motor forces. In figure 2a the block diagram of a servo loop is shown.

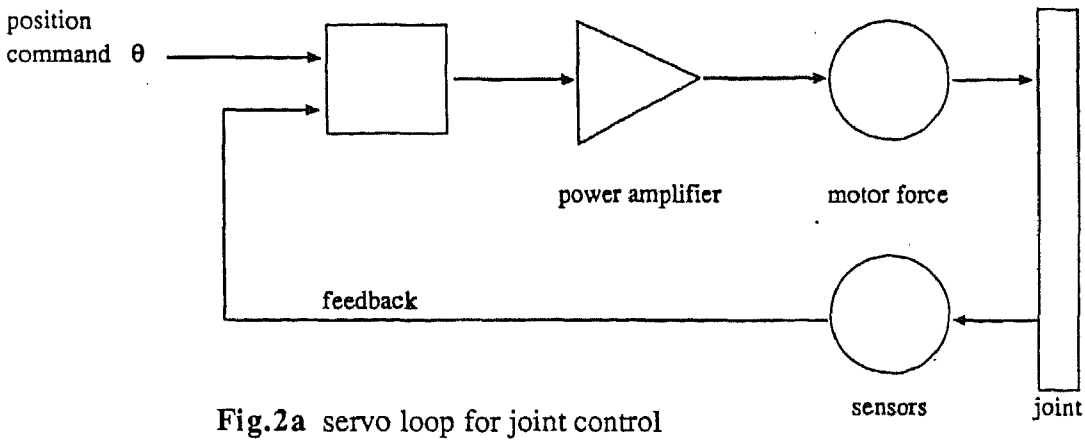


Fig.2a servo loop for joint control

The base coordinate system (*world coordinates*) in a work cell is typically a cartesian coordinate system, whereas the position of the joints are measured in joint coordinates, e.g. angles.
Since the positioning commands are fed to the servo loop in real time, there is not enough time for the transformation of joint coordinates into cartesian coordinates for servo control purposes. For this reason the servo loop is often implemented in joint coordinates, leaving it to an compiler or interpreter of the list of positioning commands to do the conversion work in advance and to produce the list of joint coordinates. This approach hinders the developement of flexible, mobile robots.
Let us regard the positioning problem now a little bit closer.
For the transformation of the position of an object in the coordinate system *m* of the manipulator and end-effector ("robot hand") into the world coordinates *w*

$$\mathbf{x}_m \text{ (joint position } \Theta) \mapsto \mathbf{x}_w \qquad \Theta = (\theta_1,\theta_2,\theta_3)^T$$

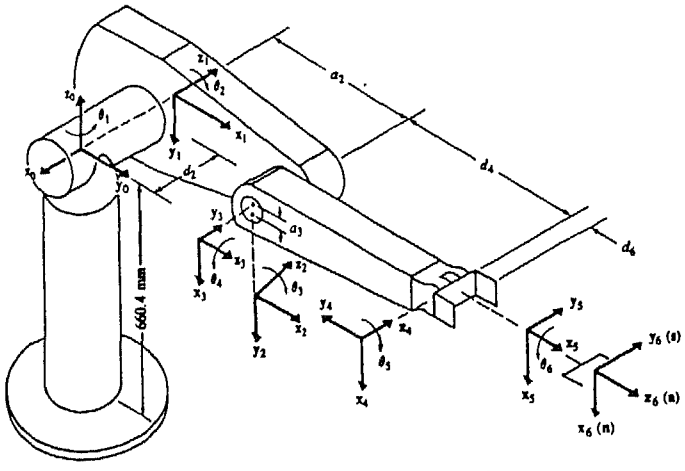
a good solution is provided by the so-called *homogen transformation*

$$\mathbf{x}_w = T(\Theta)\, \mathbf{x}_m$$

with the augmented cartesian position $\mathbf{x} = (x_1,x_2,x_3,\ 1)^T$ and the 4x4 transformation matrix $T(\Theta)$ which comprises the effect of a 3-dim rotation and translation.
Denavit and Hartenberg [DEN] showed that the whole transformation for a manipulator with N joints can be done also by a sequence of N single transformations $T_1, \ldots, T_N$, each one associated with the transformation of the coordinate system of one joint of the manipulator

$$T(\Theta) = {}^0T_1\ {}^1T_2 \ldots {}^{N-1}T_N$$

In figure 2b a manipulator of the PUMA robot type is shown with its joint coordinate systems.



PUMA robot arm link coordinate parameters					
Joint i	θ_i	α_i	a_i	d_i	Joint range
1	90	−90	0	0	−160 to +160
2	0	0	431.8 mm	149.09 mm	−225 to 45
3	90	90	−20.32 mm	0	−45 to 225
4	0	−90	0	433.07 mm	−110 to 170
5	0	90	0	0	−100 to 100
6	0	0	0	56.25 mm	−266 to 266

Fig.2b The PUMA robot manipulator, from [FU]

As we see, the problem to transform hand coordinates to world coordinates can be directly solved, once the joint coordinates are given.

Unfortunately, the inverse problem for getting the joint position Θ when the transformation $\mathbf{T}(\Theta)$ is given (*inverse kinematic*) is not easy to solve. For arbitrary joints, there exist no standard method to obtain a closed form solution (see [FU],pp.52). To make the solution possible in closed form, designers of manipulators are motivated to orient the axes of the joints in parallel or to intersect them by 90 degree. Even then, the closed form solution is not simple and difficult to compute in real time. Additionally, there are several solutions (arm configurations) possible for one hand position. Examples are shown in figure 2c.

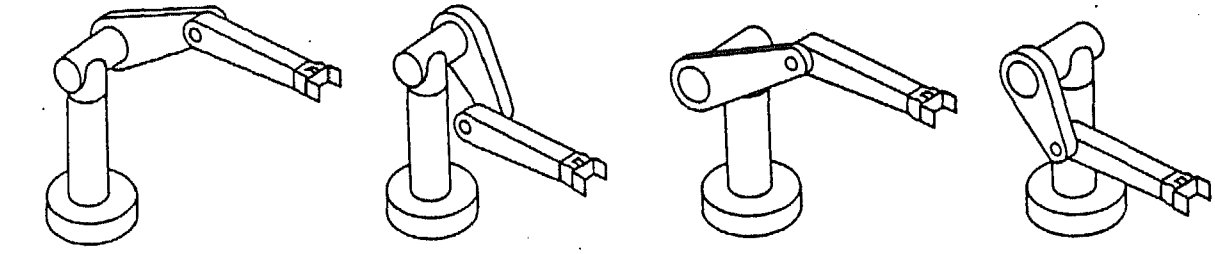


Fig 2c possible arm configurations

All these problems are avoided by the approach of learning the positioning instead of computing it. This will be described in the next section.

3) Topology conserving maps and robot control

One of the first mathematical models which exhibit topographic properties was introduced by Willshaw and v.d.Malsburg 1976 [WILL] and analyzed for instance by S.Amari 1980 [AMA]. The best known one is the one introduced by Kohonen 1982 [KOH1] or [KOH2] and analyzed for instance by Ritter and Schulten [RITT1]. Let us now briefly describe this algorithm.

Consider an input space X with the input events, characterized by data tupels $\mathbf{x} = (x_1, .. , x_p)$, x_i a real number of $\Re$ and an output space $\{\mathbf{y} = (y_1, ... , y_q)\}$ with y_i a natural number with an upper bound. So the input space is projected on an output space of descrete points $\mathbf{y}$ (*neurons*), determined by q natural numbers (indices). To each $\mathbf{y}$ of the output space there corresponds a set $\{\mathbf{x}\}$ of points (*a class*) of the input space. In figure 3a this tesselation of the input space is shown for p=q=2 . Since it is finite and bounded, the whole set of points $\{\mathbf{y}\}$ can also be ordered by one index k.

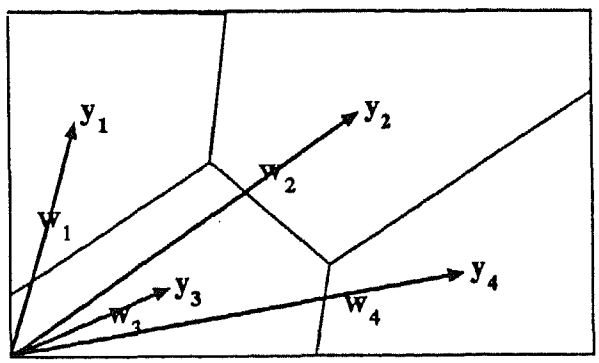


Fig 3a Tesselation of the input space by the neurons $\mathbf{y}_k$ and their weight vectors $\mathbf{w}_k$

Let every point $\mathbf{y}$ (neuron) weight the input by one weight per input component, i.e. by a weight vector $\mathbf{w} = (w_1, ... , w_p)$ from X.
Suppose, the input events $\mathbf{x}$ (t), t=1..n occur sequentially. Each one is mapped to its class $\mathbf{y}_r$ by

$$|\mathbf{x}-\mathbf{w}_c| = \min_k |\mathbf{x}-\mathbf{w}_k| \qquad (3.1)$$

This input - output mapping defines a neighbourhood of points $\mathbf{x}$ around every $\mathbf{w}_c$ to be mapped to the neuron $\mathbf{y}_c$. The following stochastic learning step for the weights has topology-conserving capabilities (see [KOH3]):

In the (t+1)-th iteration step, change the weight vector
for all neurons which are in the neighbourhood of $\mathbf{y}_c$ to

$$\mathbf{w}_k(t+1) = \mathbf{w}_k(t) + \gamma(t+1)\, h(t+1,c,k)\, [\mathbf{x}(t+1) - \mathbf{w}_k(t)] \qquad (3.2)$$

This is accomplished by the *neighbourhood function*

$$h(t, c, k) = \begin{cases} 1 & \text{if } \mathbf{y}_k \text{ is in the neighbourhood } N_c(t) \text{ of } \mathbf{y}_c \\ 0 & \text{else} \end{cases}$$

and the conditions for the *learning rate* $\gamma(t)$

$$\lim_{t\to\infty} \gamma(t) = 0, \quad \sum_{t=1}^{\infty} \gamma(t) > \infty , \quad \sum_{t=1}^{\infty} \gamma(t)^2 < \infty \qquad (3.3)$$

The difference of this stochastic algorithm, minimizing the least mean square error (LSME), to the classic ones (see e.g. [TOU]), lies in the definition of a neighbourhood for the learning process. In the classic case, either all weights (*class prototypes*) are updated (which cause fluctuations in one part of the mapping to pass to other, more distant parts) or only one weight (the selected class prototype) is updated, resulting in a poor convergence of the weights of rare selected neurons. In figure 3b a sequence of converging states of the mapping of a set of 2-dim inputs to a 2-dim neural network is shown. In the rectangle of the 2-dim input space the set of weight vectors {**w**} is drawn, each one connected with its nearest 4 neural neighbours; thus forming a 2-dim grid. The neural network itself is not shown.

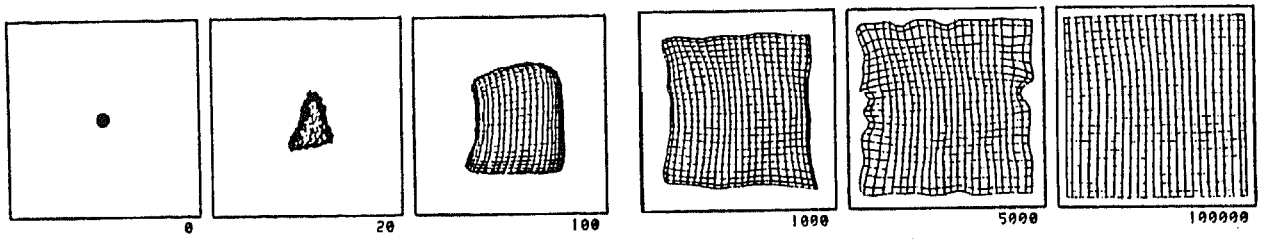


Fig 3b learning of a 2-dim topographic mapping (from [KOH3])

As we can see, the random chosen initial values of the weight vector (first picture with iteration count 0) are properly adapted reflecting the ordered, 2-dim topology of the input distribution (last picture, after 100000 iterations).

How can such a kind of mapping be used for robot control ?

In section 2 we have seen that the inverse kinematic problem is hard to solve analytically. Now, the learning algorithm (3.2) enables us to learn the mapping.
Let the sensor input space $X \subset \Re^3$ be the Cartesian space and {(i,j,k)/ i,j,k from 1..N} the grid space of the indexed neurons (see fig.5a). Then the mapping of the sensor space (perhaps deformed by sensor characteristics) to the Cartesian space is done by $\mathbf{x} \mapsto \mathbf{y}_c=(i,j,k)$ with

$$|\mathbf{x}-\mathbf{w}_c| = \min_k |\mathbf{x}-\mathbf{w}_k| \qquad (3.4)$$

To each Cartesian position $\mathbf{y}_c=(i,j,k)$ there corresponds by a non-linear mapping a joint coordinate position Θ_c which also should be learned.
Let $\mathbf{x}_F$ denote the final position, measured after the movement by some external or internal sensor in Cartesian coordinates; i.e. joint sensor coordinates are transformed prior by $\mathbf{T}(\Theta)$.
The learning algorithm for the inverse kinematics contains therefore two learning steps:

a) For the mapping (input space → Cartesian space) take equation (3.2)

$$\mathbf{w}_k(t+1) = \mathbf{w}_k(t) + \gamma(t+1)\, h(t+1, c,k)\, [\mathbf{x}(t+1) - \mathbf{w}_k(t)] \qquad (3.5)$$

b) For the learning of the proper joint angles Θ_k corresponding to the neuron y_k take

$$\Theta_k(t+1) = \Theta_k(t) + \gamma(t+1)\, h(t+1, c,k)\, [\Theta_k^*(t+1) - \Theta_k(t)] \qquad (3.6)$$

The neighbourhood function h(.) can be varied; for instance Ritter and Schulten [RITT3] assumed h(.) to be a Gaußian-shaped function, e.g. $h(t,c,k) := \exp(-(y_c-y_k)^2 / 2\sigma(t)^2)$, instead of a step function. In both cases, the neighbourhood is made smaller with increasing t by decreasing the step-width or the standard deviation σ of the Gaußian distribution.
Since we map a real-valued position $\mathbf{x}$ to an indexed position $y_c=(i,j,k)$ with a certain Θ_c, we get a positional error (see section 5.1). To reduce this resolution error, we approximate the true position $\Theta_{true}(\mathbf{x})$ by the sum of the coarse resolution value Θ_c and a linear approximation $\Delta\Theta = \mathbf{A}\,(\mathbf{x}-\mathbf{w})$, the first term of a Taylor expansion:

$$\Theta(\mathbf{x}) = \Theta_c + \Delta\Theta = \Theta_c + \mathbf{A}_c\,(\mathbf{x}-\mathbf{w}_c) \qquad (3.7)$$

Certainly, the matrix $\mathbf{A}_c$ is a good approximation only for a small section of the output space and is therefore different for different positions (i,j,k). Following Ritter, Martinetz and Schulten [RITT3], we first make a coarse positioning, get the sensored, real position $\mathbf{x}_I$, and then make the fine movement with (3.7) and measure finally the resulting position $\mathbf{x}_F$.
All the coefficients of Θ_c and $\mathbf{A}_c$ can be put into a general parameter vector contains 12 components: the 3 joint coordinates of Θ_c and the 9 matrix coefficients $A_{11}, \ldots, A_{33}$:

$$\mathbf{u}_c = (\theta_1,\theta_2,\theta_3,A_{11}, \ldots ,A_{33})^T{}_c$$

The learning of the set of parameters for coarse and fine movement replace the rule (3.6) by a learning rule for the general parameter vector $\mathbf{u}_c(n)$:

$$\mathbf{u}_c(t+1) = \mathbf{u}_c(t) + h(.)\gamma(t+1)[\mathbf{u}_c^*(t+1)-\mathbf{u}_c(t)] \qquad (3.8)$$

with the neighbourhood-function h(.) of (3.7) and the $(t+1)^{th}$ estimation $\mathbf{u}_c^*$ of $\mathbf{u}_c$.

What are good estimations of Θ_c^* and $\mathbf{A}_c^*$?

The new estimation of Θ_c is obtained by using the measured error $(\mathbf{x}-\mathbf{x}_F)$ in the linear approximation of (3.7)

$$\Theta_c^* = \Theta_c + \mathbf{A}_c(\mathbf{x}-\mathbf{x}_F) \qquad (3.9)$$

The new estimation of $\mathbf{A}_c$ uses both the measured positions $\mathbf{x}_I$ and $\mathbf{x}_F$:

$$\mathbf{A}_c^* = \mathbf{A}_c + \mathbf{A}_c(\,(\mathbf{x}-\mathbf{x}_F) - (\mathbf{w}_c-\mathbf{x}_I)\,)\,(\mathbf{x}_F-\mathbf{x}_I)^T / |(\mathbf{x}_F-\mathbf{x}_I)|^2 \qquad (3.10)$$

It should be noticed that an estimation which is more easy to calculate and which does not use an intermediate positioning $\mathbf{x}_I$ is given by

$$(\mathbf{A}_c^*)_{ij} := [\theta_i(\mathbf{x}_F+d\mathbf{x}) - \theta_i(\mathbf{x}_F)]/dx_j = [\mathbf{A}\,d\mathbf{x}]_i / dx_j \approx [\mathbf{A}(\mathbf{x}-\mathbf{x}_F)]_i / (\mathbf{x}-\mathbf{x}_F)_j \qquad (3.11)$$

which uses the fact that $\mathbf{A}$ is the first derivation in the first term of the Taylor expansion. Since $\mathbf{x}-\mathbf{x}_F$ has the expectation value of zero$_c$ in the cell ijk, the estimator $\mathbf{A}_c^*$ is unbiased.

4.) Optimal mappings and maximal information gain

Let us now consider the characteristics of an *optimal* mapping.
This leads us to the question : *optimal - in what sense?*
Let us consider a mapping as it is shown for example in figure 3a. Since sets of points of the input space are mapped to single points in the output space, there is certainly less information in the input than in the output. One plausible principle of a good mapping is to transmit as much information from the input to the output as possible (*maximal information principle*). This optimality criterion was recently proposed by Linsker [LIN1], who suggested that this might be a fundamental principle for the organization of biological neural systems.
Knowing the input pattern $\mathbf{x}$, the Shannon information gain from the N output points $\mathbf{w}_i$ is

$$I_{trans} = I_{out} - I_{out/inp} = -\ln[P(\mathbf{w}_i)] + \ln[P(\mathbf{w}_i/\mathbf{x})]$$

The average transmitted information for all inputs and outputs is with the expectation operation

$$\langle f(\mathbf{w}_i)\rangle := \Sigma_{\mathbf{w}_i} P(\mathbf{w}_i)\, f(\mathbf{w}_i)$$

$$\langle I_{trans}\rangle_{\mathbf{w}_i,\mathbf{x}} = \langle I_{out}\rangle_{\mathbf{w}_i,\mathbf{x}} - \langle I_{out/inp}\rangle_{\mathbf{w}_i,\mathbf{x}} = -\Sigma_i P(\mathbf{w}_i)\ln[P(\mathbf{w}_i)] \; - \; \Sigma_{\mathbf{x}} P(\mathbf{x}) \Sigma_i P(\mathbf{w}_i/\mathbf{x})\ln[P(\mathbf{w}_i/\mathbf{x})]$$

The average transmitted information $\langle I_{trans}\rangle$ is maximized when

$$\langle I_{out}\rangle_{\mathbf{w}_i,\mathbf{x}} \stackrel{!}{=} \max \qquad (4.1)$$

$$\langle I_{out/inp}\rangle_{\mathbf{w}_i,\mathbf{x}} \stackrel{!}{=} \min \qquad (4.2)$$

It is easy to see by variation analysis in appendix A that (4.1) is satisfied when

$$P(\mathbf{w}_i) = P(\mathbf{w}_j) = 1/N \qquad \text{for all } i,j \qquad (4.3)$$

For the demand (4.2) we know that the values for $P(\mathbf{w}_i/\mathbf{x})$ must be very unequal to yield a minimum. This is fulfilled for a tesselation of the output space, as indicated in picture 3a, without specifying *how* it was obtained, for instance by a mapping like the one of equation (3.1). Specifically, every input pattern $\mathbf{x}$ is only assigned to one appropriate class $\mathbf{y}_i$.
Then we have

for all $\mathbf{x}$ of $\mathbf{w}_i$ $\quad P(\mathbf{w}_i/\mathbf{x})\ln[P(\mathbf{w}_i/\mathbf{x})] = 1 \ln[1] = 0$

for all $\mathbf{x}$ not of $\mathbf{w}_i$ $\quad P(\mathbf{w}_i/\mathbf{x})\ln[P(\mathbf{w}_i/\mathbf{x})] = \lim_{P\to 0} P \ln[P] = \lim_{P\to 0} \frac{(\ln[P])'}{(1/P)'} = \lim_{P\to 0} -P = 0$

and therefore

$$\langle I_{out/inp}\rangle = 0$$

This means, that for a maximal average information transmission it is sufficient to have

$$P(\mathbf{w}_i) = 1/N.$$

What does this mean for the *density of the classes* (number of classes per input space area unit, also called *magnification factor*) in the input space ?
The class density is identical to the point density of the class prototypes $\mathbf{w}_i$. In the optimal mapping every class has the same occurence probability 1/N and therefore the number K of classes in a certain area ΔA of the input space is

$$K := \frac{\text{probability mass of the whole area } \Delta A}{\text{average probability of one class}} = \int_{\Delta A} p(\mathbf{x})\, d\mathbf{x} \;/\; 1/N$$

With the number of classes per area $K/\Delta A$ the class density or *magnification factor* $M(\mathbf{x})$ becomes

$$M(\mathbf{x}) = \lim_{\Delta A \to 0} K/\Delta A = \lim_{\Delta A \to 0} N/\Delta A \int_{\Delta A} p(\mathbf{x})\, d\mathbf{x} = N\, p(\mathbf{x}) \qquad (4.4)$$

In other words, for the topology conserving mapping which preserves the maximum of information *the point density of the class prototypes must approximate the probability distribution of the input patterns*.
It should be noted that this is contrary to the findings of Linsker himself in [LIN2], who stated that in optimal topology-conserving maps the often referenced classes should become bigger in the space, not smaller.
For robot control this demand is quite instructive to interprete. If we have regions of the action space where the action occur very often, this region should be better controlled and should have therefore a better resolution to minimize the average control error as it is shown in figure 1b.

Is this demand satisfied for the topology-conserving maps introduced in section 3 ?
As we know from equation (3.1) and the considerations before, equation (4.2) is satisfied.
Additionally, Kohonen found in [KOH2] for a one-dimensional array of class prototypes that their point density converge to the input distribution. Contrary to this, Ritter and Schulten found by calculating the n-dimensional case [RITT1] that this is not true, but that the magnification factor is proportional to $p(\mathbf{x})^{2/3}$, for the 2-dim (complex) case they also found $M(\mathbf{x}) \sim p(\mathbf{x})$. Therefore, at least for the 2-dim case, Kohonens mapping fulfills equation (4.4) and so (4.3) and (4.1) and can be termed *optimal*.

5) Error analysis of the non-linear mapping

Since we map an infinite set of real-valued input events to a fixed number of discrete positions (i,j,k), we have a positional error. Certainly, the more positions (i,j,k) we have, the smaller the error will be; but there is always a principal error. Even with a linear approximation (see section 3) the resulting error will be smaller, but not zero.
In this context two questions arise:

- *W hat is the principal error we make by using the topology-conserving mapping ?*
- *W hat is principal error we make by using the linear approximation ?*

For these considerations we focus our analysis on the stationary state, i.e. the mapping is learned (has converged) and do not change any more. Furthermore, let us assume that the input events are equal distributed in the Cartesian space, i.e. we do not have areas of special interest (cf. fig. 1b and section 4).
In practical applications it is more important to know the maximal possible error than the average error. So we will focus our investigations on the maximal eror of the learned mapping.

5.1 The error of the topology-conserving mapping

Let us consider a tesselation of the Cartesian input space, as shown in figure 5a. By the mapping decision of (3.1) in the stationary state the input space of equally distributed events is devided into regular cubes of edge lengthes Δx_1, Δx_2 and Δx_3.

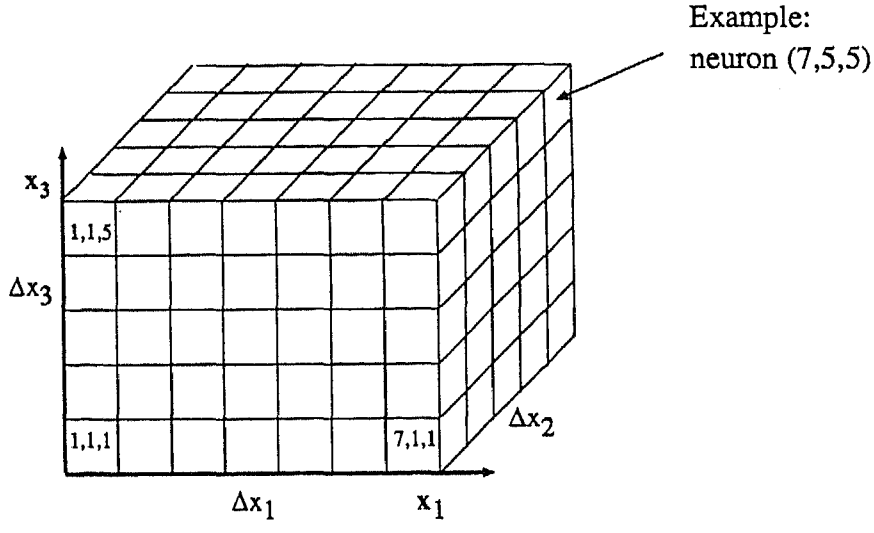


Fig. 5a The input space tesselation by the output space (i,j,k)

If the working space has the edge-lengths X_1,X_2 and X_3 then the space contains $N=n_1n_2n_3$ cubes (or neurons) with $n_i := X_i / \Delta x_i$. The maximal deviation of the correct positioning occurs obviously on the boarder of the the classes.
The *maximal positioning error* in a regular grid is therefore

$$e^{max} = | \Delta x /2 | = 1/2 \, (\Delta x_1^2 + \Delta x_2^2 + \Delta x_3^2)^{1/2} \qquad (5.1)$$

Example: For a cubic workspace with the edgelength of 70 cm and N=1000 neurons you have an error of 7 $x3^{1/2}$=12.12 cm which is much too high for normal robot operation.

5.2 The error of the linear approximation

The error of the coarse movement of section 5.1 may be corrected by the introduction of an additional fine movement, i.e. a linear approximation. Let us now compare the error we make by this linear approximation of the exact analytical solution for the inverse kinematic problem.
Since the analytical solution is different for different types of robots, let us regard the commonly used PUMA manipulator type as shown in figure 2b. It has three joint angles $\theta_1,\theta_2,\theta_3$ and the constant length a_2,a_3,d_2,d_4 for the arm movement and another triple joints $\theta_4,\theta_5,\theta_6$ for the hand movement (which do not concern us for the moment).
Let us regard the error of the arm movement, i.e. positioning the hand, characterized by the vector p from the base to the intersection of the last three joint axes, see figure 5b.

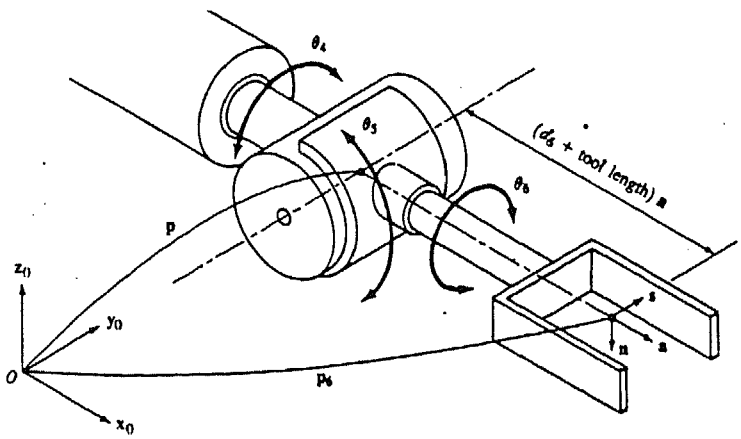


Fig.5b Definition of the position **p** (from [FU],p.43)

According to [FU], pp.63, the position $\mathbf{p} = (p_1,p_2,p_3)^T$ of the manipulator hand in (shifted) world coordinates has the following angles as solutions:

$$\begin{aligned} \theta_1 &= \tan^{-1}(f_1(p_1,p_2,d_2)) &&:= t_1(\mathbf{p}) \quad |\theta_1| \le \pi \\ \theta_2 &= \tan^{-1}(f_2(p_1,p_2,p_3,\ a_2,a_3,d_2d_4)) &&:= t_2(\mathbf{p}) \quad |\theta_2| \le \pi \\ \theta_3 &= \tan^{-1}(f_3(p_1,p_2,p_3,\ a_2,a_3,d_2d_4)) &&:= t_3(\mathbf{p}) \quad |\theta_3| \le \pi \end{aligned} \tag{5.2}$$

The three functions $t_1(.)$, $t_2(.)$ and $t_3(.)$ are shown in figure 5c.

Let us now take a closer look to the goal of the stochastic approximation of section 3.
When the algorithm has converged, we know that

$$(\Theta_{true}(w_c))_i = t_i(w_c) = (\Theta_c)_i \tag{5.3}$$

i.e. the estimation for the joint coordinates of w_c has converged to the true value. Then the matrix A_c has converged, too.

$$\theta_1 = \tan^{-1}\left[\frac{-\text{ARM}\, p_y\sqrt{p_x^2+p_y^2-d_2^2} - p_x d_2}{-\text{ARM}\, p_x\sqrt{p_x^2+p_y^2-d_2^2} + p_y d_2}\right] \qquad -\pi \leq \theta_1 \leq \pi$$

Arm configurations	ARM	ELBOW
LEFT and ABOVE arm	−1	+1
LEFT and BELOW arm	−1	−1
RIGHT and ABOVE arm	+1	+1
RIGHT and BELOW arm	+1	−1

$$\theta_2 = \tan^{-1}\left[\frac{\sin\alpha\cos\beta + (\text{ARM}\cdot\text{ELBOW})\cos\alpha\sin\beta}{\cos\alpha\cos\beta - (\text{ARM}\cdot\text{ELBOW})\sin\alpha\sin\beta}\right] \qquad -\pi \leq \theta_2 \leq \pi$$

$$\cos\alpha = -\frac{\text{ARM}\cdot\sqrt{p_x^2+p_y^2-d_2^2}}{\sqrt{p_x^2+p_y^2+p_z^2-d_2^2}} \qquad \cos\beta = \frac{p_x^2+p_y^2+p_z^2+a_1^2-d_2^2-(d_4^2+a_3^2)}{2a_2\sqrt{p_x^2+p_y^2+p_z^2-d_2^2}}$$

$$\sin\alpha = -\frac{p_z}{\sqrt{p_x^2+p_y^2+p_z^2-d_2^2}} \qquad \sin\beta = \sqrt{1-\cos^2\beta}$$

$$\theta_3 = \tan^{-1}\left[\frac{\sin\phi\cos\beta - \cos\phi\sin\beta}{\cos\phi\cos\beta + \sin\phi\sin\beta}\right] \qquad -\pi \leq \theta_3 \leq \pi.$$

$$\sin\phi = \text{ARM}\cdot\text{ELBOW}\sqrt{1-\cos^2\phi} \qquad \cos\phi = \frac{a_2^2+(d_4^2+a_3^2)-R^2}{2a_2\sqrt{d_4^2+a_3^2}}$$

$$\sin\beta = \frac{d_4}{\sqrt{d_4^2+a_3^2}} \qquad \cos\beta = \frac{|a_3|}{\sqrt{d_4^2+a_3^2}} \qquad R = \sqrt{p_x^2+p_y^2+p_z^2-d_2^2}$$

Fig. 5c The exact solutions for the PUMA inverse kinematics

The linear approximation is visualized in figure 5d for one dimension. As we can see, for a non-linear function the maximal error of the approximation is obtained at the boarder of a neuron-controlled cube-cell. For a constant probability distribution of input events the stochastic approximation of (3.6) minimizes the quadratic error which corresponds to the amount of space between the hyperplane of the approximation and the function surface **t(x)** in the small region around Θ_c.

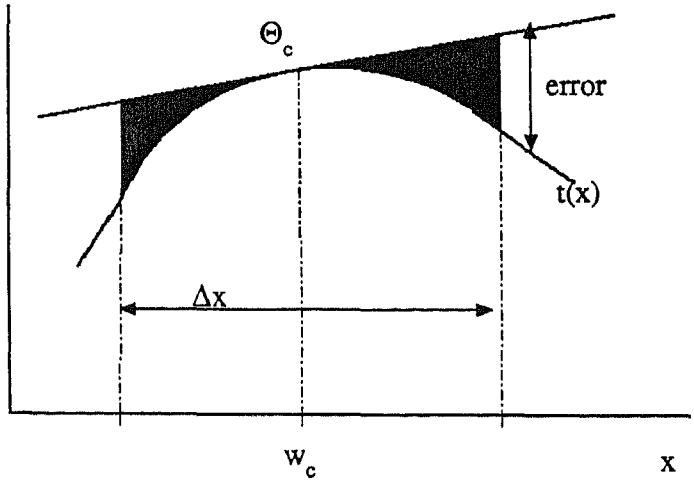


Fig. 5d The error of the linear approximation

In the case of many neurons i.e. small cell regions we can assume that **t(x)** behaves well enough such that the stochastically approximated hyperplane (represented by $\mathbf{A}_c$) can be substituted by the tangential hyperplane at Θ_c, i.e. the learned matrix A really represents the first derivate in the Taylor expansion of the true position in equation (3.7). This also assumes that the error is at all cell boarders approximately the same.

Then the matrix values $(A_{ij})_c$ of the matrix $\mathbf{A}_c$ can easily obtained by using only small variations dx_i (developement of $\mathbf{t}(\mathbf{w}_c+dx_i)$ around $\mathbf{w}_c$) as the quotient of differences.
Calculating this in every vector component gives us

$$A_{ij} \approx [t_i (w_c+dx_j/2) - t_i (w_c-dx_j/2)] / dx_j \qquad (5.4)$$

To show the typical error problems of the neural network approach let us define a linear path in the workspace. In figure 5e such a path is shown in a cubic workspace, denoted with START and END.

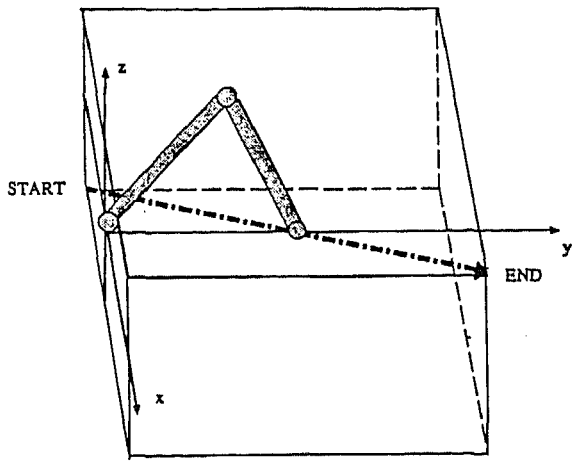


Fig. 5e A linear path in the workspace

With the knowledge of equations (5.2) and (5.4) we can calculate the maximal position error in the joint space for the descrete case of a PUMA robot manipulator for a workspace length of 717mm.
Let us first regard the error which is made in each joint. In figure 5f left the absolute values of the three angles $\theta_1,\theta_2,\theta_3$ and in the right figure the relative errors

$$e_i^{MAX} := (\Delta t_i(\Delta x/2) - \Delta\theta_i(\Delta x/2)) / t_i(w_c)$$

are shown.

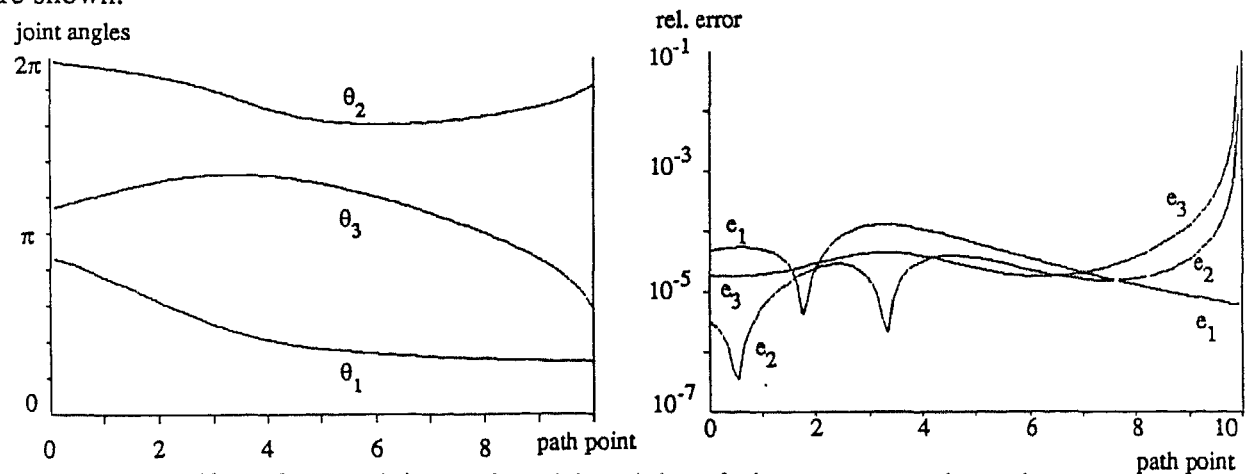


Fig. 5f The joint angles $t_i(\mathbf{x})$ and the relative error e_i on the path

As we can see, the three functions $t_i(\mathbf{x})$ on the left hand side behave quite smoothly; the sharp peaks of minima in the plot of the relative error e_i (shown on a logarithmic scale) on the right hand side

indicate us regions of good linearity of $t_i(\mathbf{x})$. The angles (or points in the workspace) are characterized by an approximately constant first and zero second derivate; the peaks are thus located at the turnpoints of the $t_i(\mathbf{x})$.

Nevertheless, the order of the absolute, maximal joint error is determined by the joint angle with the maximal deviation which itself is mainly determined by the number of neurons which have to control the work space. This dependance is shown in in figure 5g as the computed absolute error as a function of the path position in the cubic workspace. Parameter is the resolution n, the number of neurons in one dimension of the neuron grid (i,j,k).

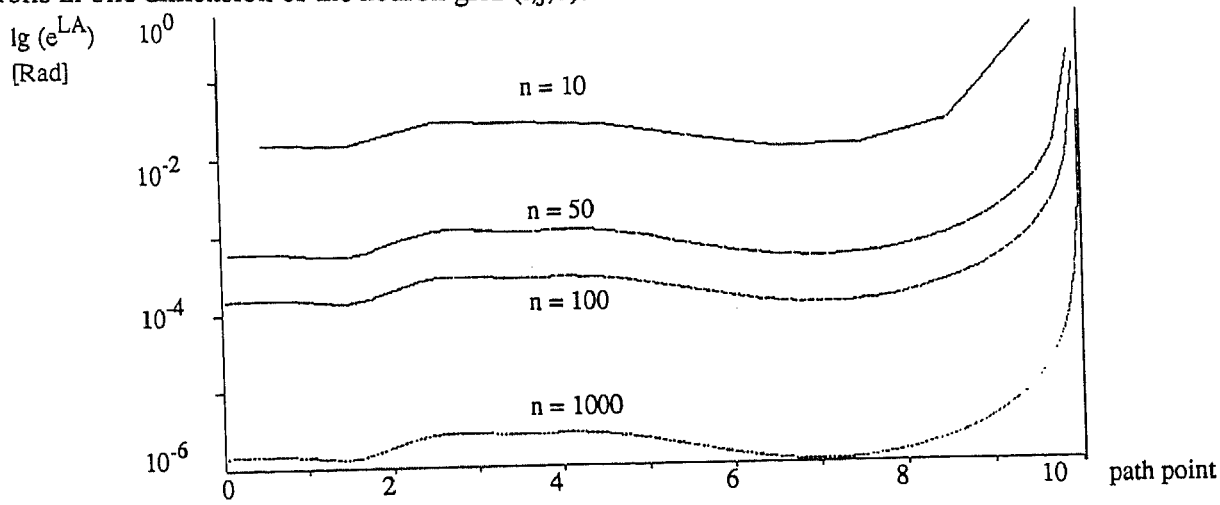


Fig. 5g The absolute joint space error as a function of the resolution

As one can expect, the error increases when coarse resolution is used. The error of the linear approximated angle is very dependant of the position in the path.

Nevertheless, for applications the maximal positioning error in the Cartesian space is more important. By

$$e_{cart}^{LA} := | \mathbf{x} - \mathbf{x}^* | \tag{5.5}$$

with the approximated position

$$\mathbf{x}^* = {}^0\mathbf{T}_4(\Theta)\,\mathbf{a}$$

which is the transformation of the hand position of the linearly approximated angle Θ of (3.7)by the PUMA transformation matrix (see section 2) we have according to [FU,p.63]

$$\mathbf{x}^* = \begin{pmatrix} C_1(a_2C_2 + a_3C_{23} + d_4S_{23}) - d_2S_1 \\ S_1(a_2C_2 + a_3C_{23} + d_4S_{23}) + d_2C_1 \\ d_4C_{23} - a_3S_{23} - a_2S_2 \end{pmatrix} \tag{5.6}$$

with $C_i = \cos(\theta_i)$, $S_i = \sin(\theta_i)$, $C_{ij} = \cos(\theta_i+\theta_j)$, $S_{ij} = \sin(\theta_i+\theta_j)$

To compute the error we assume again that the algorithm has converged and the position has been learned. Then we can compute $\mathbf{x}^*$ for a $\mathbf{x}$ on a class boarder just by computing Θ (using equations (3.7) and (5.4) and the equations of figure 5c for Θ_c) and applying (5.6) on it.

The positional error due to the joint angle approximation in the Cartesian space as a function of the control space resolution is shown in figure 5h.

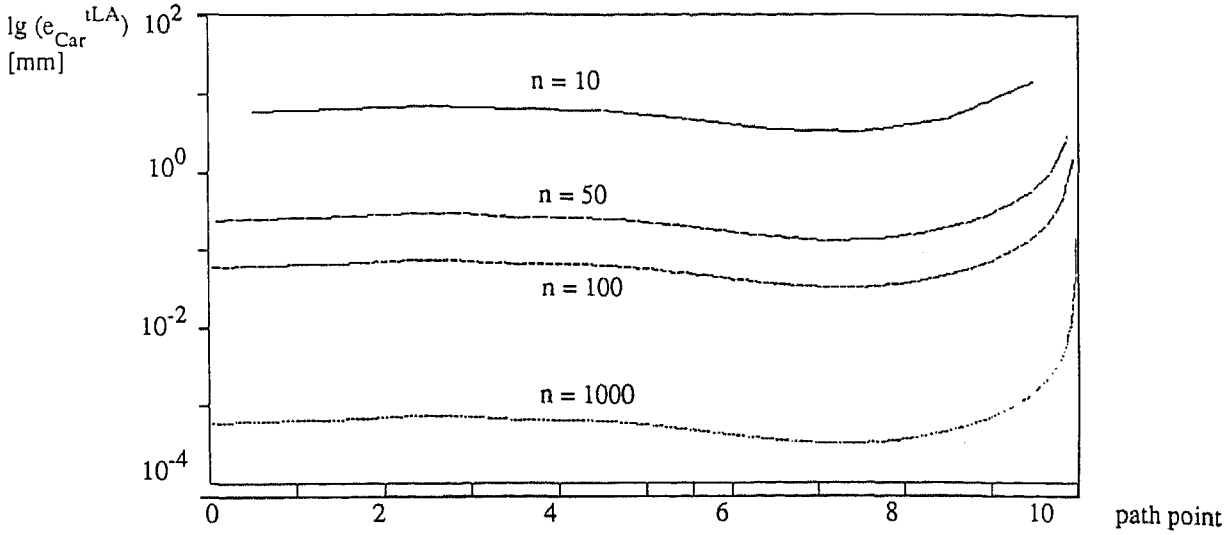


Fig.5h The absolute positional error as a function of the number of neurons

Certainly, the error increases with coarse resolution of the neuronal net, too. We also can see that the error in the angle space transforms non-linearly to the Cartesian space. Like in figure 5g, the error of the manipulator is much greater at the boarders of the workspace than in the middle. It is therefore a good conventional practice to cut out the actual workspace from the possible workspace to avoid positioning errors at the boarders. Nevertheless, in the neural network approach this is not really necessary: if we use certain areas in the physical workspace very often, for instance when we transfer loads to a destination point to charge another machine, the topological representation of this area will be inreased and the positioning resolution becomes finer there, decreasing the positioning error.

When we regard figures 5g and 5h, we notice that the functions for n=10,100,1000 seem to be the same in one figure, only shifted for a certain, constant amount. Thus, the logarithm of the error of the linear approximation $\lg(e^{LA})$ should be linear in the logarithm of n :

$$\lg(e^{LA}) \sim -\lg(n) \quad \text{or} \quad \lg(e^{LA}) = a + b \lg(n), \quad b<0 \tag{5.7}$$

This gives the function

$$e^{LA} = C\, n^b \qquad \text{with } C := 10^a \tag{5.8}$$

In figure 5i left the relation between $\lg(e^{LA})$ to $\lg(n)$ is shown for one point P of the linear path (the local maximum of error in the first part of the path). This function can be approximated very effectively by a linear relation, giving us for our PUMA robot values of b = -2.65672 and C = 2.01329 .

The approximation error can be seen as a kind of resolution error of the network due to the finite, limited number of neurons. If we set the approximation error e^{LA} equal to the one of an hypothetical discretization error which is at most half of the discretization increment we have

$$2\, e^{LA} =: \text{inc}(r_n) = \text{value range } V_\theta \text{ of theta / number of states } 2^{r_n}$$

and we define a minimal approximation resolution r_n of the network

$$r_n = \mathrm{ld}\,(V_\theta / e^{LA}) - 1 \qquad V_\theta = 2\pi \tag{5.9}$$

The approximation resolution r_n on the position P in the path is shown in figure 5i on the right hand side. As we can suggest from equations (5.11) and (5.9) we have $r_n \sim -\mathrm{ld}(e^{LA}) \sim \lg(n)$ which is reflected in the right figure 5i .

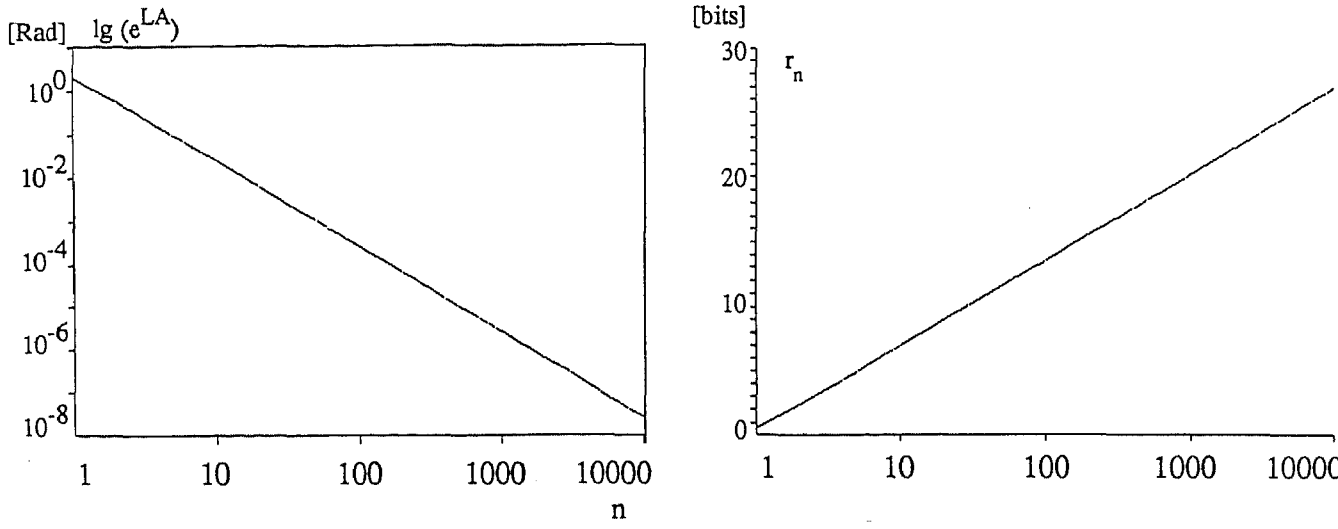


Fig 5i The absolute joint error and the resolution r_n as a function of the number of neurons in one dimension

6.) The trade-off between performance and storage size

The real time performance of the robot control algorithm by topology-conserving maps is quite good, because the non-linear mapping is done essentially by the lookup-tables (which are learned) and not by real-time calculations.
Nevertheless, even when the learning overhead is small (the learning of all parameters is done and the mapping is stable), we have to pay a price for the fast control: the price of a big storage size. The better the positioning resolution is, the more storage for the lookup-tables is required.
Let us briefly calculate the necessary amount of storage for a given positioning resolution.

6.1 Constant resolution

Assuming a workspace of $X_1=X_2=X_3= 71.7$ cm length a stored number of 12 bit resolution gives us an resolution increment (error) of 0.175 mm; a 10 bit resolution gives only 0.7 mm resolution.
Since our system is specified for each "neuron" by 3 weights of w_c, 3 joint coordinates Θ_c and 9 matrix coefficients of A_c we have for $N = n_1 n_2 n_3 = n^3$ neurons with the same resolution of $r = r_w = r_\theta = r_A$ bits in each number (storage element) and a necessary storage of

$$s = n^3 (3r_w+3r_\theta+9r_A) \quad \text{Bits} \qquad (6.1)$$

or $$n^3 (3+3+9) = 15\, n^3 \quad \text{storage elements (SE)}$$

With the network parameter n we get the following table of storage requirements

n / neurons N	10 / 10^3	50 / $1.25\ 10^5$	100 / 10^6	1000 / 10^9
number of SE	$1.5\ 10^4$	$1.87\ 10^6$	$1.5\ 10^7$	$1.5\ 10^{10}$
3SE = 4byte (10bit res.)	20kB	2.5MB	20MB	20GB
2SE = 4byte (12bit res.)	30kB	3.74MB	30MB	30GB

Table 6a Resolution and storage requirements

As we can see, a good spacial control resolution (high n) is closely related to high storage requirements. Since the necessary storage is a function of the order $O(n^3)$ of the spacial control resolution, the practical application of the algorithm is limited in the present implementation stage by the storage requirements.

It should be noted that this calculation is independant whether the algorithm is implemented in VLSI hardware by neuron-like structures or merely simulated on a conventional computer system.

6.2 Positioning error and optimal resolution

In the previous section (Table 6a) we have seen that a high resolution of the Cartesian position leads to big storage requirements. Therefore we have to revise carefully the storage needs for effective neural network control. Now, given a descrete application with a maximal tolerated positioning error, how much storage amount have to be provided?
In our neural network control system we have two kinds of errors due to resolutions

- the coordinate resolution error (maximal 1/2 digitalization increment) due to the digitalization process of the real values which represent a coordinate in the joint or Cartesian space

- the neural network resolution error due to the linear approximation of the joint angles

It is clear that it is not feasable to choose a high coordinate resolution with a small error and a low network resolution yielding an high approximation error or vice versa. Since there rests always a principal error in each coordinate being approximated, the increase in digital resolution of the coordinate position value does not increase the accuracy of the approximation as well.

Let us evaluate now the relation between the storage size and the maximal error made by the linear approximated position with finite resolution. By this evaluation, we hope to get some hints how to choose the neural network parameter n and the resolutions r_w,r_θ and r_A of the variables w_c, $\Theta_c$and A_c which determine the mapping in equations (3.4) and (3.7). Since we assume a fault-free transformation of joint angles to Cartesian coordinates by the robot manipulator mechanics, the Cartesian error can always directly be calculated using (5.6) when the error in the joints are given.

The overall maximal positioning error is therefore determined by the superposition of two independant sources of error:

$$e^{MAX} = | e^{LA} + e^{RES} | \tag{6.2}$$

For a certain storage increment Δs the error will change by

$$\Delta e^{MAX}(s) = \frac{d}{ds} e^{MAX}(s) \quad \Delta s \tag{6.3}$$

Let us assume that we take some storage amount from one kind of variable and put it to another one, i.e we change two resolution parameters without changing the overall storage requirements.
Let h(s-Δs) and g(s+Δs) be the errors of the two kind of variables after the change in the storage configuration. The error $e^{MAX}(s)$ will then change to $\underline{e}^{MAX}(s)$ by the first order approximation

$$\begin{aligned} & e^{MAX}(g(s),h(s)) + [\frac{\partial}{\partial g} e^{MAX}(g) \frac{\partial}{\partial s} g(s) - \frac{\partial}{\partial h} e^{MAX}(h) \frac{\partial}{\partial s} h(s)] \Delta s && \textit{reduce storage in h} \\ & = e^{MAX}(g(s),h(s-\Delta s)) - [\frac{\partial}{\partial g} e^{MAX}(h) \frac{\partial}{\partial s} g(s)] \Delta s && \textit{add the storage to g} \\ & = e^{MAX}(g(s+\Delta s),h(s-\Delta s)) =: \underline{e}^{MAX}(s) \end{aligned}$$

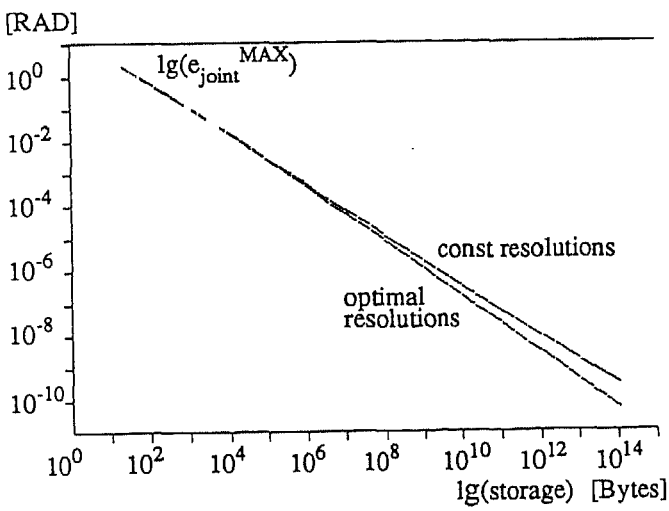
The error will therefore diminish when

$$\frac{\partial}{\partial g} e^{MAX}(g) \; \frac{\partial}{\partial s} g(s) \; < \; \frac{\partial}{\partial h} e^{MAX}(h) \; \frac{\partial}{\partial s} h(s)$$

If the two derivates are equal, no storage rearrangement can diminish the error any more. The storage configuration can therefore be termed *optimal.*
This idea can be applied to the multi-variable case. For the developement of (6.2) we have

$$\Delta e^{MAX}(s) = [\frac{\partial}{\partial n} e^{MAX}(n) \frac{\partial}{\partial s} n(s) + \frac{\partial}{\partial r_w} e^{MAX}(r_w) \frac{\partial}{\partial s} r_w(s) + \frac{\partial}{\partial r_\theta} e^{MAX}(r_\theta) \frac{\partial}{\partial s} r_\theta(s) + \frac{\partial}{\partial r_A} e^{MAX}(r_A) \frac{\partial}{\partial s} r_A(s)] \; \Delta s \tag{6.4}$$

For an optimal storage configuration all terms should have equal values. This leads us to a system of three equations with the four variables n, r_w, r_θ and r_A. In appendix B this is solved, getting three variables as a function of the forth. By additionally using the storage equation (6.1) we finally can calculate the maximal joint positioning error $e^{MAX}(s_{opt})$ as a function of the optimal storage requirement s_{opt}. This is plotted in figure 6.2a for the point P in the linear path (cf. fig.5i). In figure 6.2b the corresponding maximal Cartesian error is shown. For comparison, in the same plots the errors using optimized n, but equal resolutions $r_w=r_\theta=r_A=: r$ are additionally shown.



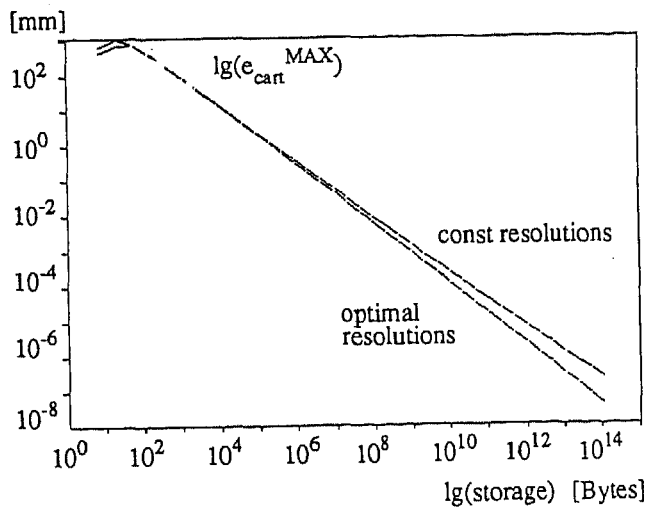


Fig. 6.2 a) The maximal joint error of optimal storage

b) The corresponding Cartesian error

We can see, that there is really a difference between the optimized distribution of resolutions and the non-optimized, constant one. In the case of the Cartesian error at 1.13 10^{14} Bytes storage, the error due to non-optimized resolutions is 5.6 times greater than the optimized one!
Nevertheless, if we regard the configuration with an Cartesian error of 0.201 mm, a value which is in the range of normal mechanical inaccuracy and therefore more important for practical applications, the necessary 1.9 MB of storage memory is contained in 39.6^3 neurons with a resolution of r = 16.4 Bits. The optimal configuration (r_w=17, r_θ=20.1, r_A=15 Bits, see fig. B.1a) gives an error of 0.164 mm, only 18% less than the non-optimized one!

Therefore, if the software problem of using floating point calculations with different numbers of bits is considered, it seems not advisable for practical simulations and for microprocessor control of robots to use different resolutions for the different storage variables $\mathbf{w}_c$, Θ_c and $\mathbf{A}_c$.

7. Discussion

In this paper it was shown how the topology-conserving mappings can be used for the control of robot manipulators. Furthermore, an optimal mapping in the sense of maximal information transmission results when the magnification factor of the pattern space tesselation equals the probability density of the pattern distribution.
The approach of describing the inverse kinematics by a set of stored function values and learning them by executing the positioning task reveals some interesting properties:

- ♣ The inverse control is very fast because it is based on a memory mapping and not on analytical calculations using transcendental functions.

- ♣ There are no analytical solutions necessary. This provides an easy control even of multi-joint manipulators with worn-out joints.

- ♣ The learning algorithm provides a better resolution for often used regions of interest and enables the introduction of positioning restrictions.

Nevertheless, the method of learning a mapping provides also some problems.

- ● One of it consists of the time overhead for the updating algorithm. It must be underlined that the algorithm presented in section 3 is essentially a sequential one since it uses a global decision (3.1) for searching the neuron with the minimal distance. It should be noted that the algorithm can be parallelized as it was shown by Kohonen in [KOH3]. This feature can be exploited by multiprocessor systems or, more effective, by neural chips which model each neuron by a separate hardware unit, thus representing a fast, adequate hardware base for the parallel algorithm.

- ● Another problem is the high amount of storage necessary for the memory mapping. As it was shown in this paper, an optimized storage approach can overcome this problem and reduce the storage amount for reasonable positioning errors to the modest request of less than 2 MBytes.

Additionally, some problems of robot manipulator control should be mentioned which still rest to be solved:

- The neural positioning represents only an approach for the low level primitives which are used by higher layers such as trajectory generation which in turn is used by movement generation.
  The low level approach is completely isolated in respect to the higher level functions and is not applicable to them.

- The neural positioning is only learned for a fixed workspace. If the workspace changes by an affine transformation, i.e. a translation, a rotation or a scaling, the mapping is no longer valid and must be relearned.
  The topology-conserving memory mapping can be regarded as a special case of an associative memory, with all its adjacent problems.

- Time sequences of positionings can not be used on other start positions as the original one in contrast to human beings who can repeat the same learned movement on different start positions: There is no "abstract", position independant coding of a movement.

In summary, the topology-conserving memory mapping can be regarded as an interesting, new approach for the problem of inverse kinematics which promises good results in practical applications. Nevertheless, there rest some important problems to be solved for a satisfactory theory of robot movement control.

References

[AMA] S. Amari, Topographic Organization of Nerve Fields, Bulletin of Math. Biol., 1980, Vol 42, pp 339-364

[DEN] J. Denavit, R.S. Hartenberg, A Kinematic Notation for Lower-Pair Mechanismen Based on Matrices, Journal Appl. Mech., 1955, Vol 77, pp.215-221

[FU] K.S.Fu, R.C. Gonzales, C.S.G. Lee, Robotics, McGraw Hill, 1987

[KOH1] T. Kohonen, Self-organized Formation of Topologically Correct Feature Maps, Biological Cybernetics, 1982, Vol 43, pp 59-69

[KOH2] T. Kohonen, Clustering, Taxonomy and topological Maps of Patterns, IEEE Proc. 6th Int. Conf. Pattern Recognition, Oct. 1982, pp.114-128

[KOH3] T. Kohonen, Self Organization and Associative Memory, Springer Verlag 1984

[LIN1] R.Linsker, Self-Organization in a Perceptual Network, IEEE Computer, March 1988, pp.105-117

[LIN2] R.Linsker, Towards an Organizing Principle for a layered Perceptual Network, in D. Anderson (ed), Neural Information Processing Systems, Amer. Inst. of Physics (NY), 1988

[RITT1] H.Ritter, K.Schulten, On the Stationary State of Kohonen's Self-Organizing Sensory Mapping, Biological Cybernetics, 1986, Vol 54, pp.99-106

[RITT2] H.Ritter, K.Schulten, Convergence Properties of Kohonen's Topology Conserving Maps, Biological Cybernetics (in press)

[RITT3] H.Ritter, T. Martinetz, K. Schulten, Topology-Conserving Maps for Learning Visuo-Motor-Coordination, Neural Networks, Vol 2/3 , pp. 159-167, Pergamon Press 1989, New York

[SCH] R.F.Schmidt, G.Thews, Einführung in die Physiologie des Menschen, Springer Verlag, Berlin 1976

[TOU] Tou, Gonzales, Pattern Recognition Principles, Addison-Wesley 1974

[WILL] Willshaw, v.d. Malsburg, How Patterned Neural Connections can be set up by Self-Organization, Proc. Royal Soc. (1976), Vol 194, pp.431-445

Appendix A: The maximal expected information

Theorem: The expected information $E := \sum^N_{i=1} P_i \ln(P_i)$ is maximal, if $P_i = P_j =: P = 1/N$

Proof: Let P_i be arbitrary functions of a parameter *t*. Then E is a maximum, if

$$\frac{dE}{dt} = \frac{d}{dt}\left(\sum_{i=1}^{N} P_i(t) \ln(P_i(t)) \right) = \sum_{i=1}^{N} \frac{\partial}{\partial P_i} [\, P_i \ln(P_i) \,] \frac{dP_i(t)}{dt}$$

$$= \sum_{i=1}^{N} [\, \ln(P_i) + 1 \,] \frac{dP_i}{dt} = 0 \qquad \text{(A.1)}$$

With the restriction

$$\sum_{i=1}^{N} P_i = 1 \qquad \text{(A.2)} \qquad \text{or} \qquad \sum_{i=1}^{N} \frac{dP_i(t)}{dt} = 0 \qquad \text{(A.3)}$$

we get by adding a multiple of (A.3) to equation (A.1)

$$\frac{dE}{dt} = \sum_{i=1}^{N} [\, \ln(P_i) + a \,] \frac{dP_i}{dt} = 0 \qquad \text{(A.4)}$$

It is sufficient for this condition to be true that $ln(P_i) + a$ becomes zero or, generally spoken since *a* is arbitrary but fixed for all *i*, $\ln(P_i)$ and therefore P_i becomes independant of the index *i*.

With the condition (A.2) we get

$$\sum_{i=1}^{N} P_i = N P = 1$$

or $P = 1/N$

q.e.d.

Appendix B

The optimal parameters for minimal storage requirements

In section 6.2 equation (6.2) we computed the positioning error $e^{MAX}(s)$ of the neural network positioning approach as a superposition of two sources of error: the error of using only a linear approximation for the non-linear inverse kinematic equations and the error of the finite resolution of the learned position variables of the mapping:

$$e^{MAX}(s) = | e^{LA}(s) + e^{RES}(s) | = [\sum_{i=1}^{3} e_i^{MAX}(n,r_w,r_\theta,r_A)^2]^{1/2} \qquad (B.1)$$

with n neurons in one dimension and the resolutions r_w,r_θ and r_A for the variables w_c, θ_c and A_c.
The demand for equal terms in equation (6.4) of section 6.2 becomes therefore for (B.1)

$$\frac{\partial}{\partial s} e^{MAX}(s) = 1/2 \; [\sum_{i=1}^{3} e_i^{MAX}(n,r_w,r_\theta,r_A)^2]^{-1/2} \; 2 \sum_{i=1}^{3} e_i^{MAX}(n,r_w,r_\theta,r_A) \; (term1+term2+term3+term4)$$

with the four terms

$$\begin{aligned}
term1 &:= \frac{\partial}{\partial n} e_i^{MAX}(n) \frac{\partial n}{\partial s} = \frac{\partial}{\partial n} (e_i^{LA}(n) + e_i^{RES}(n)) \frac{\partial n}{\partial s} \qquad (B.2)\\
term2 &:= \frac{\partial}{\partial r_w} e_i^{MAX}(r_w) \frac{\partial r_w}{\partial s} = \frac{\partial}{\partial r_w} e_i^{RES}(n,r_w,r_\theta,r_A) \frac{\partial r_w}{\partial s}\\
term3 &:= \frac{\partial}{\partial r_\theta} e_i^{MAX}(r_\theta) \frac{\partial r_\theta}{\partial s} = \frac{\partial}{\partial r_\theta} e_i^{RES}(n,r_w,r_\theta,r_A) \frac{\partial r_\theta}{\partial s}\\
term4 &:= \frac{\partial}{\partial r_A} e_i^{MAX}(r_A) \frac{\partial r_A}{\partial s} = \frac{\partial}{\partial r_A} e_i^{RES}(n,r_w,r_\theta,r_A) \frac{\partial r_A}{\partial s}
\end{aligned}$$

A sufficient condition for the equality of the terms in equation (6.4) is the equality of the four terms of (B.2). This gives us three conditions for four parameters. Thus, e^{MAX} depends on just one parameter in the optimal storage configuration.
Now, let us explicitely calculate these dependancies by calculating the four terms of (B.2).
Since the storage function $s(n,r_w,r_\theta,r_A)$ of equation (6.1) has an inverse one, we can write

$$\partial x/\partial s = (\partial s/\partial x)^{-1} \qquad \text{and therefore} \qquad \begin{aligned} \partial n/\partial s &= [3n^2\, 3(r_w+r_\theta+3r_A)]^{-1} \\ \partial r_w/\partial s &= [n^3\, 3]^{-1} \\ \partial r_\theta/\partial s &= [n^3\, 3]^{-1} \\ \partial r_A/\partial s &= [n^3\, 9]^{-1} \end{aligned} \qquad (B.3)$$

The resolution error is determined by the resolution errors in equation (3.7)

$$\Theta + e^{RES} = \Theta + \delta\Theta = (\Theta_c + \delta\Theta_c) + (A_c+\delta A_c)\,((w_c+\delta w_c)-x)$$

or

$$e^{RES} = \delta\Theta = \delta\Theta_c + \delta A_c(w_c-x) + A_c\delta w_c + \delta A_c\delta w_c$$

The maximal error occurs at the boarder of the cell y_c=(i,j,k) (see figures 5a and 5d) and when the value of a variable differs by half of the increment of that variable. This means for every component of the vector

$$\max (w\text{-}x) = \Delta x/2 = X/(2n) \qquad \text{and} \qquad \delta z = 1/2 \ inc_z$$

and so

$$e_i^{RES} = 1/2\, inc_\theta + 3/2 \ inc_A \, X/(2n) + 1/2 \ inc_w \sum_{j=1}^{3} A_{ij} + 3/2 \ inc_A \ 1/2\, inc_w \qquad (B.4)$$

Since we can not presume any information about the possible values (states) of a variable z, the encoding of the variable by r bits uses uniform probability density or constant increments. Thus, we can define the increment of z as the range V_z devided by the number of possible states of the variable:

$$inc_z := V_z / 2^r \qquad (B.5)$$

The corresponding values for our problem are

$V_\theta = 2\pi = 6.2831854 \ldots$
$V_A = 2.0\ 10^{-4}$ on the linear path of section 5
$V_w = X = 7.17\ 10^4 \quad [10^{-2}\ mm]$

$S_A := \max_i \sum_{j=1}^{3} A_{ij} = 1.0\ 10^{-5}$ for the PUMA configuration in point P

Note: By this approximation e_i^{MAX} becomes independant of the index i, resulting in $e^{MAX}(s) = 3^{1/2}\, |e_i^{MAX}|$

Now we can compute the terms. With equations (5.8) and (B.3) we get from (B.2) (B.6)

term1: $\frac{\partial}{\partial n}(e_i^{LA}(n) + e_i^{RES}(n)) \frac{\partial n}{\partial s} = (C\, n^{b-1} - \frac{X}{2n^2} \frac{3}{2} inc_A)\ [3n^2\, 3(r_w + r_\theta + 3r_A)]^{-1}$

$\frac{\partial}{\partial r} inc_z = \frac{\partial}{\partial r} V_z\, 2^{-r} = -\ln(2)\, inc_z$

term2: $\frac{\partial}{\partial r_w} e_i^{RES}(r_w) \frac{\partial r_w}{\partial s} = -1/2\ \ln(2)\, (inc_w S_A + 3\ inc_A\ 1/2\, inc_w)\ [n^3\, 3]^{-1}$

term3: $\frac{\partial}{\partial r_\theta} e_i^{RES}(r_\theta) \frac{\partial r_\theta}{\partial s} = -1/2\ \ln(2)\, inc_\theta\, [n^3\, 3]^{-1}$

term4: $\frac{\partial}{\partial r_A} e_i^{RES}(r_A) \frac{\partial r_A}{\partial s} = -1/2\ \ln(2)\, inc_A\, 3\, (X/(2n) + 1/2\, inc_w)\, [n^3\, 9]^{-1}$

With the demand term4 $\stackrel{!}{=}$ term2 we get

$$inc_A\, X/(2n) = inc_w S_A + inc_A\, inc_w$$

or

$$inc_A = inc_w S_A / (X/(2n) - inc_w) \qquad (B.7)$$

With the demand term3 $\stackrel{!}{=}$ term2 we get

$$inc_\theta = inc_w S_A + 3\ inc_A\ 1/2\, inc_w = inc_w\, (S_A + 3/2\ inc_A) \qquad (B.8)$$

With the demand term3 $\overset{!}{=}$ term1 we get

$$-1/2\ \ln(2)\ inc_\theta\ [n^3\ 3]^{-1} = (C\ n^{b-1} - \underset{2n^2}{X}\ \frac{3}{2}\ inc_A)\ [3n^2\ 3(r_w+r_\theta+3r_A)]^{-1} \qquad / [n^3\ 3]$$

$$R := (r_w+r_\theta+3r_A)$$

$$-1/2\ \ln(2)\ inc_\theta = (C\ n^b - \underset{2n}{X}\ \frac{3}{2}\ inc_A)\ [3\ R]^{-1}$$

$$g(n_{opt},r_w) := X/2n_{opt}\ 3/2\ inc_A - C\ n^b{}_{opt}\ -3/2\ \ln(2)\ inc_\theta\ R = 0$$

The function $g(n,r_w)$ is difficult to solve analytically. Therefore, to get the value of n_{opt} when r_w is given, the problem of calculating the zero-crossing of g(.) is solved by a Newton iteration. The resulting values for n,r_θ and r_A when r_w is given is plotted against $s(n,r_w,r_\theta,r_A)$ in figure B1.

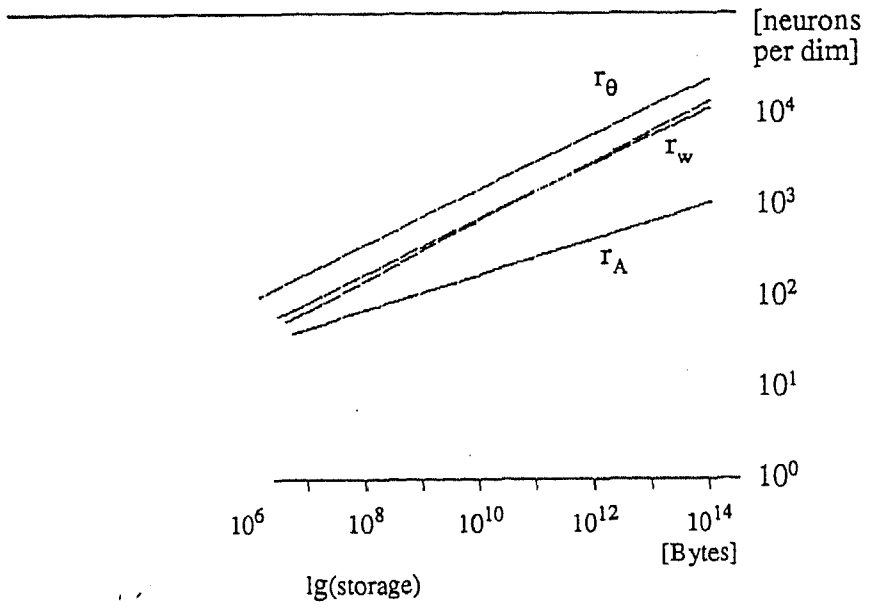


Fig.B1 The parameters for optimal storage