UPTEC X 06 039
AUG 2006

ISSN 1401-2138

KRISTOFFER FORSLUND

Reconstruction of dynamic mRNA networks

Master’s degree project

UPPSALA UNIVERSITET

Bioinformatics Programme

Uppsala University School of Engineering

UPTEC X 06 039	**Date of issue 2006-09**

Author

Kristoffer Forslund

Title (English)

Reconstruction of dynamic mRNA networks

Title (Swedish)

Abstract

This work uses computer simulations to evaluate algorithms that analyze gene regulatory networks by systematically perturbed gene expression measurements. To investigate the accuracy of a family of such algorithms, they are applied repeatedly to a variety of simulated gene regulatory networks over a range of conditions. Thereby, statistical comparisons may be made between different methods. Specifically, an algorithm called Mode of action by Network Identification (MNI) is shown to be at least comparable to its parent method, and is shown to be able to distinguish between direct and indirect target genes for a given treatment under noise-free circumstances. To further evaluate the performance of MNI, further research might apply it to a wider range of existing expression data collections.

Keywords

Gene regulation, MNI, system recovery, simulation, mRNA, perturbation method

Supervisors

Mats Gustafsson
Computational medicine group

Scientific reviewer

Rolf Larsson
Institutionen för klinisk farmakologi

Project name	Sponsors
Language **English**	Security
ISSN 1401-2138	Classification
Supplementary bibliographical information	Pages **68**

Biology Education Centre Biomedical Center Husargatan 3 Uppsala
Box 592 S-75124 Uppsala Tel +46 (0)18 4710000 Fax +46 (0)18 555217

Reconstruction of dynamic mRNA networks

Kristoffer Forslund

Sammanfattning

För att kunna förstå människor eller andra biologiska system behöver vi veta vad deras gener gör. Detta innebär dels deras kemiska funktion, och dels de villkor under vilka de aktiveras. Hur olika gener aktiverar eller avaktiverar varandra, det vi kallar det genetiska reglernätverket, är en av de centrala frågorna i den moderna biologin, både för att vi vill förstå organismerna i sig själva och för att kunna inverka på dem. Exempelvis dyker ofta situationen upp att ett läkemedel, kanske en cancerbehandling, har effekt men utan att vi är säkra på vilken gen den faktiskt inverkar på. För att kunna besvara sådana frågor behöver vi ett sätt att kartlägga det genetiska reglernätverket.

Det har lagts fram metoder för att försöka göra detta. Dessa utgår från experiment där man systematiskt stör celler av den typ man vill studera genom att överuttrycka specifika gener. Därefter mäts hur mycket mRNA som produceras för varje gen och resultaten sätts samman till en datamängd som beskriver hur mRNA-nivåerna relaterar till varandra under olika villkor. Denna datamängd analyseras sedan bioinformatiskt, genom en algoritm som söker det reglernätverk som bäst motsvarar experimenten. Hur väl de här algoritmerna faktiskt fungerar är inte helt klarlagt.

Detta arbete prövar ut sådana algoritmer för analys av genetiska reglernätverk. Detta görs genom att simulera ett stort antal experiment på virtuella system av gener under olika omständigheter. Därigenom kan algoritmens resultat jämföras direkt med de faktiska system som användes för simulationen, och på så vis kan dess effektivitet utvärderas. Här har fokuserats på två specifika sådana algoritmer, och det visas att de fungerar i princip.

Examensarbete 20 p i Bioinformatikprogrammet

Uppsala universitet augusti 2006

Table of Contents

1 Introduction

1.1 Overview

This section (1) begins with an overview of its contents (1.1), then proceeds (1.2) by describing the background and the underlying set of biological questions that the techniques addressed in this thesis seeks to answer. Next follows a very brief introduction to microarray technology (1.3) and to pattern recognition (1.4) as they relate to this problem. The goals (1.5) of the project are described, and finally (1.6) the report as a whole is outlined.

1.2 Background

In the so-called "post-genomic" era following the completion of the Human Genome Project, biology has increasingly shifted towards holistic or systemic approaches, giving rise to the buzzwords of genomics, proteomics, peptidomics, transcriptomics etc. These are all variations on a central theme; as we now have access to almost every single component of the (human) biological make-up, and increasingly powerful methods of measuring several quantities at once, we may begin to study on a large scale how these components interact. A core idea here is that biological responses cannot be adequatly modelled without considering the setting in which they take place; understanding of a disease or molecular mechanism does not come from studying a single gene, but rather a set of interacting genes that together with the surrounding circumstances form a whole.

Another key theme to the post-genomic era is that after the human genome sequencing provided a map of our genetic make-up, the post-genomic research will provide an understanding of that map. Determining the functions of the 30,000 or so genes we possess is an enormous task. Because of the magnitude of this endeavour, as well as the holistic understanding mentioned previously, systematic approaches must be taken here, entailing significant bioinformatic efforts. By studying how several genes interact, it may be possible to draw conclusions on their respective function; moreover, the process may to an increasing degree be automated ([5], [14], [4], [15], [27]).

Much of this work concerns gene regulation; that is, understanding not merely what chemical or otherwise catalytic properties a gene product has, but why, when and where it is expressed and as a result of what stimuli. The properties are, of course, connected, so that it may be possible to draw chemical conclusions concerning a protein by studying what situations cause it to be up- or down-regulated and which its interactory partners are; or conversely, draw conclusions on activating circumstances due to known catalytic function. Furthermore, it stands clear that a full understanding of the gene regulatory network of an organism would enable extensive predictions of what stimulus could produce what effect, having significant consequences for treating medical conditions or - if desired - changing how that organism functions and develops ([5], [14], [4], [15], [27]).

One important medical area where gene regulatory information becomes particularly important is that of cancer treatment. Depending on the particularities of a given cancerous growth it may or may not respond to particular therapies. Determining what drugs are successful in a given case may be a lengthy and costly process, and if it could be performed easier, it may mean that adequate care can become more readily available. When a cancer is resistant to standard therapies, often a screening assay is attempted, in order to find a suitable alternative. If gene interactions could be more easily determined, this might not be necessary, as then a direct gene target could be selected for that particular cancer based on such understanding. Conversely, understanding of such regulatory networks would enable clarification of what the exact targets of a given therapy or compound is, increasing our understanding of what to use where. Such knowledge of drug mechanisms and targets would also help us direct targeted research to handle, say, a variant cancer which is resistant to standard treatments.

All of these reasons make the development of general techniques for large-scale surveying of gene regulatory networks important. Lately, approaches combining large-scale gene expression studies, using microarray (see below) or quantitative polymerase chain reaction (qPCR) techniques with bioinformatics have been suggested by some authors for such surveying. These perturbation techniques build on causing several artificial externally induced changes to the gene expression patterns of the target systems, then estimating the actual regulatory interactions from the reactions to such artificial changes. Perturbations may include treating cell culture with various drugs, or adding inducible plasmids to directly alter the expression of some mRNAs, or more drastic or complex

methods such as gene deletions or RNA interference. ([9], [7], [24], [2], [14], [23]).

1.3 Microarrays

To measure levels of gene expression, one readily accessible method is that of the microarray. Using polymerase chain reaction techniques, the messenger RNA content of a cell sample may be converted to a sample, similar in relative concentrations, of DNA, a cDNA sample. Use of specially tagged nucleotides in this step enables making the cDNA sample fluorescent or otherwise making it easier to measure its concentration. In basic microarray methodology (two-channel system), this is done both for a perturbed system and an unperturbed reference case, using different fluorescent dyes. For the next step, a microarray chip is used. This is a surface with single-strand DNA sequences attached to it in discrete spots corresponding to subsequences of specific genes, usually many thousands, spanning large parts of the genome of an organism. Both dye-tagged cDNA samples (perturbed sample and reference) are added to the microarray and allowed to hybridize to the immobilized single-strand nucleotide sequences in the spots. Fluorescence intensity is then measured for both dyes (both channels) and the ratio of intensities will correspond to the relative abundance of each transcript between perturbed sample and baseline, and thus yield a measure of gene expression under those particular conditions. Single-channel techniques, such as the Affymetrix array, allow measurement of absolute transcript concentrations under ideal circumstances ([26]).

1.4 Data analysis and pattern recognition

Measuring the perturbed levels of gene expression across the system, techniques ranging from simple numeric analysis to sophisticated machine learning or artificial intelligence approaches are taken to estimate the interdependencies between expression levels. Thus - it is hoped - this should yield a mechanistic understanding of the system ([9], [7], [24], [2], [14], [23]).

There are, however, several reasons to be cautious about the claims of such methods. There are many hazards to pattern recognition. The basic problem setup in this field is as follows: based on a series of observations, which we

call the *training set* or the *training experiments*, we seek to adapt a model of a given complexity so that it can predict not merely the training set, but any set of observations made of the same real-world system. Since no measurement of physical reality is ever perfectly exact, all such measurements are subject to *measurement noise*. The more complex a model we choose - the more parameters it contains, the more complex types of *system dynamics* or behaviours can it describe. However, unless we have a training set which is larger than the minimum needed to determine the model parameters, the resulting model will vary significantly with even small variations in the training set - in effect, we will model the aforementioned measurement noise along with the actual system dynamics. This is a form of what is known as *over-fitting*. While this may in part be remedied by increasing the number of experiments underlying the analysis, it might in practice only be possible to measure so much, leading to an ever-present trade-off between not drawing incorrect conclusions because of over-fitting and being able to capture interesting aspects of the behaviour of the system ([28], [25]).

From this point of view, a systematic evaluation of perturbation methods is clearly needed. The current work may be seen as a precursor to such an evaluation, investigating the ability of a particular set of perturbation methods to retrieve correctly the interdependencies of simulated gene regulatory networks. See ([12], [28], [21], [22], [25], [5]) for more information on simulated gene networks.

1.5 Project goals

The goal of this thesis is to determine the performance of perturbation methods like the MNI approach put forward by di Bernardo et al ([7]) by applying it to simulated systems based on that of Zak et al ([28]). To avoid jumping to conclusions, the perturbation strategies are evaluated over a wide range of simulated network models under different conditions. This partly avoids the potential risk that a particular algorithm overperforms the others substantially in the context of one net and one condition, while performing poorly in other cases and conditions.

Another aspect of the study is to examine how much the performance of MNI and NIR depend on the "built-in" choices of model complexity that implicitly or

explicitly result from their various assumptions of degree of interactions between genes with regards to their transcrips. This is referred to as the *connectivity* of the models and defined as the maximum number of genes whose expression independently may affect the expression of any given gene; in effect, how many intra-transcriptional influences any gene may be subject to.

1.6 Outline of this thesis

This first section (1) describes background and project goals, as well as an introduction to project and report as a whole. The methods section (2) describes the algorithms tested, the simulation schemes used to do so, and the experiments performed by applying the former to the latter. The results of these experiments are described in the following section (3), after which the results are summarized, evaluated and discussed (4). The final sections are acknowledgements (5) and source references (5). Source code listings for the MATLAB environment have been omitted for space reasons, but are available from the author on request.

2 Methods, materials and algorithms

2.1 Overview

This section (2) begins with an overview (2.1) of its contents, then describes the setup and algorithms of perturbation experiment data analysis (2.2), first a simple intuitive method (2.2.1) for comparison, then both the recently published methods to be evaluated: MNI (2.2.2) and NIR (2.2.3) as well as some variants (2.2.4) of these. Other methods, which are not analyzed here, are listed in section (2.2.5). Next, the methods for evaluating (2.3) simulated data for testing purposes are presented, after which follows a comparison of the model assumptions of NIR and MNI (2.3.1) as well as their performance in published studies thus far (2.3.2, 2.3.3).

The next section describes the simulations (2.4) made in this study, the basic approach (2.4.1, 2.4.2) and the three main systems used to generate simulated datasets; the circular cascade (2.4.3), the Zak system (2.4.4), and last the random architecture system (2.4.6). Section 2.4.5 discusses how to represent the connectivity of the Zak system in matrix form. The perturbation settings (2.5) used to generate the datasets are then presented, both for single-gene perturbation experiments (2.5.1) and for experiments where pairs of genes are perturbed (2.5.2).

Following this, the next section (2.6) describes the basic experiment setup (2.6.1) and lists the types of experiments the study entails; varying the initial values in MNI (2.6.2), investigating performance of slightly different algorithm versions (2.6.3), the dependence of the methods on measurement noise (2.6.4), experiments aiming to test the ability of the methods to distinguish direct and indirect responders (2.6.5), testing dependence on an internal method parameter (2.6.6), and last applying MNI to the dataset used to test NIR as it was originally published (2.6.7).

2.2 Methods for analysing perturbation data

2.2.1 The expression change method ("subtraction method")

Obviously, the simplest way of analyzing perturbation data in the form of steady-state transcript concentrations with the goal of determining outside influences would just be to use the unperturbed steady-state concentrations as a reference and assume that any (or at least the greatest) changes in expression are most likely to result from external influence. Since such an approach cannot distinguish between direct and indirect targets of a perturbation, this will produce a large number of false positives in the form of genes incorrectly deemed to have been externally influenced. As such, this "subtraction method" forms a good reference method to compare more complex perturbation methods against. Their performance is expressed as their capacity of distinguishing primary from secondary targets.

2.2.2 The NIR method

The Network Identification by multiple Regression (NIR) method was laid forth by Gardner et al ([9]). It attempts to study network connectivity and estimate drug targets by making use not only of the actual steady state expression patterns in each perturbation experiment but also information about which gene has been perturbed and the magnitude of that perturbation. This effectively limits the useful experiments to cases where the exact size of the perturbation can be measured. In particular, it requires controllable plasmid insertions where an additional marker gene is also expressed, and which can be measured to determine plasmid activity.

A simplified model of transcript concentration interdependency is assumed. First and most importantly, the system is reduced from the 'true' system of transcripts, translation products, protein dimers, external ligands and promotor-transcription factor complexes to a system consisting only of mRNA transcripts. This already means a great change in what kind of dynamics may be modelled. In this reduced model space, transcript changes per time unit are modelled as functions of transcript concentrations, yielding a system of differential equations.

We define

$$x_i = \frac{RNA_{i,perturbed}}{RNA_{i,unperturbed}} - 1 \tag{1}$$

as the expression change for gene i in a given experiment. Let u_i be the external influence on gene i in this experiment, and a_{ij} the influence of transcript j on transcript i. Then the model for experiment $l = 1..M$ on a system of $j = 1..N$ genes becomes

$$\frac{d}{dt}x_{il} = \sum_{j=1..N} a_{ij}x_{jl} + u_{il} \tag{2}$$

In steady state ($\frac{d}{dt}x_{il} = 0$), this can be expressed in matrix form as

$$A_{ij} = a_{ij}, X_{il} = x_{il}, P_{il} = u_{il} \tag{3}$$

$$AX = -P \tag{4}$$

With both the *expression data matrix* X and the *external influence* or *perturbation matrix* P known, this becomes at first sight a standard computational problem - solving the linear equation system for the unknown *connectivity matrix* A. However, dimensionality is an issue; the number of experiments is unlikely to be any larger than the number of genes and may in fact be smaller. This makes it impossible to solve for the connectivity matrix A, unless additional assumptions of constraints are introduced. NIR employs the relatively reasonable assumption that the connectivity matrix is sparse, that is, that each gene is influenced directly (as opposed to indirectly) only by a few other genes. This means that on each row of A, there are a relatively large number of zero elements corresponding to interactions that do not occur.

Under the assumption that at most k out of N genes influence the ith gene, the corresponding row of A may have at most k nonzero elements, distributed in

$\sum_{j=1..k} \binom{j}{N}$ ways (the number of ways in which up to k nonzero elements may be distributed between N positions). For relatively small values of k, such as three or four, it is computationally feasible to evaluate every such solution form for each gene/each row of A and select the solution for which the least squares error (the norm of the matrix $AX + P$, which is zero if and only if A, P and X satisfy the $AX = -P$ equation system) is minimized. Thus, the problem of finding the connectivity matrix becomes reduced to a lower-dimensional form and the risk of over-fitting decreases, while there may conceivably be a risk that the behaviour of some genes may not be modelled in the reduced net model.

In the original NIR implementation, the optimal connectivity k is determined using a relatively complex method. For values of k that a priori cannot be ruled out ($k = \{3, 4, 5, 6\}$ were chosen for the nine-gene case presented in ([9])), optimal connectivity patterns are computed and the statistical significance of these solutions were computed. The stability of the resulting models is tested, excluding connectivites for which the solution do not provide a stable system (i.e. that cannot reach steady state). For the remaining possible values of k, a number of simulation experiments are performed generating perturbation datasets for random gene regulatory network of the same size as the dataset being analyzed. NIR is then applied to these using each of the remaining possible values of k, and the recovered models are compared with the true models used to generate this simulated data. That connectivity is then selected for which the trade-off between coverage (percentage of proper connections recovered) and false positives was smallest. This approach, while rigorous, requires a significant amount of work as well as some measure of human arbitration. In the implementation used here, we instead use *cross-validation* to determine the ideal value for k. This is done as follows: for each experiment in the training set, take it out of that set and use the remaining experiments to recover the system model. Then, see how well the data from the experiment that was removed can be predicted from the model that was created from the other experiments. If the model has too high a connectivity, it will fit the training set very well but not the remaining data, and so the average fit when doing this for every experiment in the training set will be best for the optimal connectivity.

Note that the time required to evaluate $\binom{k}{N}$ possible connection patterns to find the optimal one may be considerable. There are $\binom{k}{N} = \frac{N!}{(N-k)!k!} = \frac{N \cdot (N-1) \ldots (N-k+1)}{1 \cdot 2 \ldots k}$ such patterns, and as the problem size N increases, this number grows qualitatively similarly to N^k which rapidly becomes untenable. In practice, it may be possible to use a heuristic of some sort, finding one connection at a time,

though this might not locate the truly optimal connection pattern.

For NIR, it is necessary to know the perturbation matrix beforehand. For experimental data, this is easiest done by setting up the experiments so that the magnitude of the external influence can be measured directly. For the dataset used in the NIR article, a controllable (metabolite-activated) plasmid is added to the cells in each experiment which overexpresses one gene at a time, but which also can express a marker gene. By measuring marker product concentration in cells under the same plasmid activation conditions, the magnitude of the perturbation is measured.

2.2.3 The MNI method

The Mode of action by Network Identification (MNI) algorithm was suggested by di Bernardo et al ([7]) and builds on the NIR method as presented above. The starting point is a training set consisting of steady-state mRNA concentration measurements of the system in question under a variety of perturbating conditions. These may be drug treatments, RNAi, plasmid additions, gene deletions etc., and it is neither crucial that the exact perturbation details are known, nor what genes they perturb, as long as in each case at least one gene among those for which transcript concentrations are taken changes its expression, and that these changes sufficiently span the space of transcript concentrations and the dynamics of the system.

Initially, a reduced-space differential equation system as in the NIR method is assumed. Let y_i be the mRNA concentration of gene i, let d_i be the degradation rate of transcript i, let n_{ij} be a regulation constant describing the influence of transcript j on the rate of transcription of transcript i, and let p_i be an external influence on the rate of transcription of gene i. Then

$$\frac{d}{dt}y_i = p_i \prod_j y_j^{n_{ij}} - d_i y_i \tag{5}$$

It is worth noting that a multiplicative model of this kind does not allow for a gene to be independently influenced by several others. If a parameter n_{ij} has a value different from 0 (representing a transcriptional influence in either di-

rection), then transcript j must have a nonzero concentration for transcript i to change, even if there are other parameters n_{ik} that have values significantly different from 0. While this is a correct model for systems where, say, the transcription of gene i is controlled by a protein heterodimer JK of the proteins J and K, it would not correctly model the system where the corresponding mRNAs j and k both independently increased production of I.

At first sight, this would seem to make the model vastly different from that used in NIR, to a degree that would make it impossible to simultaneously describe a system in both models. However, a number of additional assumptions are made. First, as in NIR only steady states are considered, that is, $\frac{d}{dt}y_i = 0$. A baseline case is defined where no external perturbations takes place, defining p_{i0} and y_{j0}. Next, the degradation constant d_i and the regulation constants n_{ij} are assumed to be perturbation-independent. Then

$$p_{i0}\prod_j y_{j0}^{n_{ij}} - d_i y_{i0} = p_i \prod_j y_j^{n_{ij}} - d_i y_i \tag{6}$$

$$\frac{y_j}{y_{j0}} = \frac{p_i}{p_{i0}} \prod_j \left(\frac{y_j}{y_{j0}}\right)^{n_{ij}} \tag{7}$$

which, employing the logarithm function, yields

$$b_i = \sum_j a_{ij} x_j \tag{8}$$

for $b_i = \log \frac{p_i}{p_{i0}}$, $x_i = \log \frac{y_j}{y_{j0}}$ and $a_{ij} = n_{ij}$ for $i \neq j$, $a_{ij} = n_{ij} - 1$ for $i = j$. While this may be scaled differently, it does mean that the final model used in the method is still linear in the space of logarithmed expression rate quotients. Written in matrix form, for a set of N genes and M experiments, we have

$$AX = -P \tag{9}$$

where $A_{ij} = a_{ij}$, $X_{ik} = x_i$ for the kth experiment and, similarly, P_{ik} is $-b_i$ for the kth experiment.

At first glance, this problem appears computationally impossible. The goal here is to obtain estimates of A and P only by means of the data matrix X. This obviously cannot be done without additional constraints and/or assumptions.

The perturbation matrix P is postulated to be relatively sparse; for each gene i of N genes present in M experiments, only in a few of those experiments (the set $perturbed(i)$, which together with the set $unperturbed(i)$ form the full experiment set $1..M$) is it assumed to be perturbed. The expression measurements for those experiments where gene i is directly perturbed is then $X(:, perturbed(i))$, that is, all columns in X with indices contained in $perturbed(i)$. If these experiments can be reliably selected, then the remaining experiments $X(:, unperturbed(i))$, that is, all columns in X with indices contained in $unperturbed(i)$, should correspond to the case

$$a_i X(:, unperturbed(i)) = p_i(unperturbed(i)) = 0 \tag{10}$$

where a_i and p_i are the ith rows of A and P respectively, corresponding to gene i, and $p_i(unperturbed(i))$ are the elements of p_i corresponding to experiments where gene i is not deemed to be perturbed (and which therefore display the effects of other genes on the expression of gene i in the absence of external influences). This is well-defined, and in this way it is possible to compute, row by row, a non-trivial solution for A for Equation 9. This requires determining for which experiments a gene is directly perturbed.

We observe that a gene's regulation of itself must be significant - if nothing else, its transcript degradation will depend on its own concentration. Thus diagonal elements of A are not identically zero. Effectively, for a_i, the ith row of A, one may assume any nonzero value for the ith element and solve for the rest, provided an estimate of the corresponding row of P, p_i. This is required to avoid the trivial solution of no regulation where no external perturbations are assumed.

To put this together, an initial guess of gene expression interdependencies (i.e. the connectivity matrix A) is made. As in NIR the computations are performed

separately for each gene. Under this guess, external perturbation estimates are computed and a sparsity condition is applied, that is, just as stated above, the perturbation matrix is required to be sparse, reflecting the assumption that each gene is only directly affected in a small fraction of the experiments. This condition can be more explicitly stated as follows. Let the algorithm be applied to data from M experiments on N genes. Within a given iteration of the algorithm, there is a hypothetical solution $A = A_e$ which satisfies $A_e X = -P$ to a degree for the matrix $P = P_e = -A_e^{-1} X$. Let $a_{e,i}$ and $p_{e,i}$ be the row of P_e and A_e that correspond to the ith gene. Let $a_{e,ij}$ and $p_{e,ij}$ be the elements of these rows corresponding to the jth experiment. By the sparsity condition, we assume that for gene i, experiment index j belongs in the set $unperturbed(i)$ if

$$|p_{e,i}(j)| < \sigma \cdot \max_{1 \leq l \leq M} (|p_{e,i}(l)|) \tag{11}$$

holds. The parameter $\sigma = 0.25$ was chosen by the authors of the MNI method empirically, from unpublished experiments on simulated data ([7]). That is, experiment j is assumed not to involve external perturbation of gene i if $p_{e,i}(j)$ is at most 25% of the maximal external perturbation that gene i is subjected to. As stated above, this condition is applied to select the experiments where external perturbations are assumed not to occur, after which Equation 10 is applied to determine A_e, row by row. This solution is used to recompute the external perturbation estimates P_e yet again using Equation 9, these are culled again using Equation 11, and the process is repeated until neither A_e or P_e change significantly between iterations, at which point $A = A_e$ and $P = P_e$ is taken as the final solution.

The initial connectivity guess A_{e0} appears to be relevant to the results. As this guess initially is used to calculate initial guess for external perturbations, this first step can be seen either as an initial guess at connectivity or as an initial guess at external influences. In the seminal paper for the MNI method, it is suggested that an initial guess at no inter-gene connections at all, only negative self-feedback ($P = X$, $A = -I$) would work well. Remarkably, this method is not used in the actual tests of the algoritm performed by the authors ([7]); there, instead, the initial guess for perturbations is taken as $P = (XX^T)^{-1}X$ rather than computed from an initial guess at connectivity, after which subsequent rounds of the algorithm proceeds as per the above ([8]). As will be seen below, this latter initial condition works much better.

2.2.4 Variant MNI methods

There are several immediate ways in which the MNI method could be altered and, possibly, improved. As it is, the MNI algorithm applies a sparsity assumption to the perturbation matrix P in each iteration, assuming that a gene is only externally influenced in a fraction of the experiments in the training set. As sparsity may also reasonably be assumed for the connectivity matrix A (as the number of direct influences on a gene should be limited), a logical step would be to similarly assume that only those elements in each row of A that are largest in an absolute-value sense represent actual interactions. Our new algogorithm candidate therefore applies another sparsity condition to both A_e and P_e: for each row $a_{e,i}$, $p_{e,i}$, only the largest (in the absolute value-sense) 50% of the elements represent significant connections or perturbations. This is termed the 50% *fixed-sparsity variant* of the MNI method. In the plots below, it is listed as *MNI - enforce* 50% *sparse A/P*.

In another similar variant we assume that the perturbation matrix is not only sparse but a column-permuted diagonal matrix, i.e. that no gene is perturbed in more than one experiment. This is a most limiting and artificial assumption, but as it holds in many of the simulation studies, it is a way of evaluating MNIs performance under artificially ideal assumption. We term this the *sparse P variant*. In the plots below, it is listed as *MNI - enforce sparse P*.

Most importantly, the initial guess at the connectivity matrix A at the start of the first iteration of the algorithm, A_{e0}, appears relevant. In the above section are presented two options, either that described in the MNI seminal article ([7]) or that actually used in the implementation of those authors ([8]). In this thesis, both of them were evaluated. The ($P = X$, $A = -I$) initial guess is listed in the plots below as *MNI - alternate start guess 1* while the $P = (XX^T)^{-1}X$ initial guess is listed simply as *MNI*.

We also try out an additional starting guess ($A_{e0} = -X^T(XX^T)^{-1}$, $P_{e0} = -A_{e0}X$) which, we postulate, may lie closer to the true connectivity and perturbations in that it fulfills Equation 9 already as the algorithm starts (though it need not be the optimal such solution), thus avoiding local minima or converging faster. This initial starting guess is listed in the plots below as *MNI - alternate start guess 2*.

2.2.5 Other perturbation methods

There are also entirely different approaches to analyzing perturbation data. These include boolean networks, Bayesian methods, and other approaches. Due to the results reported for MNI, investigating the performance of that family of methods has been the primary priority of this work, and therefore due to time constraints the other alternatives will not be covered here ([7], [9], [23], [19], [14], [16], [15], [27], [18], [2], [11], [24], [5]).

2.3 Evaluating the performance of perturbation methods

In their 2005 paper, di Bernardo et al ([7]) demonstrates their method purely by virtue of applying it to a composite dataset of yeast expression data and there comparing it to some standard methods of compound target detection. The recovered system model is neither presented nor compared to any known "actual" network model, which is not unexpected as the biological system investigated is large, complex and not fully understood. However, this means that this aspect of MNI performance does not seem to have been tested exhaustively.

2.3.1 Model differences

As presented above, MNI and NIR use quite different models. While both describe gene expression by a system of linear differential equations, one uses expression rates and the other logarithms of expression rates. In this work, we seek to simulate data according to some model, then retrieve it using either algorithm and compare the retrieved model with that used to generate the dataset. If the results are to be possible to compare between the algorithms, they must assume the same type of model, which they obviously do not.

Our solution is to perform all trials with slightly modified versions of the MNI and NIR algorithms. While the original methods assume different types of models, the core of the algorithm - arguably the part which we are interested in evaluating the performance of - simply solves a linear equation system of the form $AX = -P$. Further, with the increasing popularity of single-channel-type systems for expression measurement, we are interested in method performance

when data is supplied as absolute concentrations.

We simulate data according to

$$A(Y - Y_0) = AX = -P \tag{12}$$

for connectivity matrix A, perturbation matrix P, absolute steady-state transcript concentrations Y and absolute steady-state transcript concentrations Y_0 in the absence of perturbations. Matrices A, X and P then satisfy the form $AX = -P$ which is required for our adapted MNI and NIR variants, and the resulting solutions for A and P can be directly compared with the true values, enabling simultaneous evaluation of MNI and NIR on the same dataset.

2.3.2 Performance in published studies - NIR

The NIR method has been applied to simulated data with decent results ([6]). There are also published results ([9]) where it is applied to the SOS subnetwork of genes in E. coli, a system of nine genes perturbed using controllable overexpression by plasmids. In these tests, much of the network but not all is recovered, to the degree that the actual connections are actually known. See ([17]) for more information on the yeast genomic regulatory network.

A simulation study was also performed for NIR ([6]). In that study, gene networks of 100 genes with 10% sparsity were used to generate expression data using a setup similar to that of the current work, and connectivity matrix recovery was measured. Results from this were positive as to NIRs performance.

2.3.3 Performance in published studies - MNI

In the Nature Biotechnology article ([7]) where MNI is first presented, it is applied to a set of 6000 yeast gene expression levels over a total of 515 experiments. These are taken from two compendia of such results, the Hughes compendium ([13]) and the Mnaimneh data ([20]). From the combined dataset, a model for the yeast regulatory network over these genes were computed. Among the ex-

periments of the Hughes compendium were 11 promoter insertions, and the computational model was then used to calculate the corresponding perturbations for these 11 experiments, ranking each gene on how likely it was to be the direct target for that particular perturbation experiment - in effect, trying to perform a molecular mechanism study based only on expression data.

Ideally, the genes in question should rank highest, and this was the case for 9 of these 11 experiments. Furthermore, when rankings were compared with those acquired using *z-score* of expression change (a technique that takes variance into account to determine significance of the results), the true gene was consistently ranked higher with MNI than when using z-scores.

The model was also used to rank gene targets for fifteen experiments where compounds were added, nine of which have known targets. For these nine, MNI did not correctly identify the target genes, but for seven of the nine, the correct pathway was overrepresented among the fifty highest ranking genes. This is to be expected as this type of drug hardly affects transcription directly as with a promoter insertion, and although performance was not stellar, identification of the proper pathway may be useful in its own right. ([7])

2.4 Simulated systems

2.4.1 Simulation approach

This work focuses primarily on simulated data, both because it is available in unlimited amounts and because it lets us control all parameters directly. The works of di Bernardo et al. ([7]) complement this by providing a trial of MNI on a biological dataset, and as a final experiment, this study also applies MNI to the dataset used in ([9]) to evaluate the NIR method.

2.4.2 Linear system model

To simulate a system of genes, a linear model is assumed. We consider transcript concentrations y_i close to a perturbed or unperturbed steady state y_{i0} with perturbations u_i. Then

$$\frac{d}{dt}(y_i - y_{i0}) = \frac{d}{dt}x_i = \sum a_{ij}(y_j - y_{j0}) + u_i = \sum a_{ij}x_j + u_i \qquad (13)$$

which in the above matrix form becomes

$$AX = -P \qquad (14)$$

satisfying the basic form for the MNI and NIR algorithms ([9], [7]). The default unperturbed steady state $y_{i0}, i = 1..N$ can be set to arbitrary nonzero values. In the simulations, these parameters are randomized (distributed evenly between 0.5 and 1.0) for each run to avoid artifacts of steady-state choice.

Simulations are performed in the numerical programming environment MATLAB (Mathworks, Inc.) using its in-built least squares solvers to calculate X under different choices for A and P. To ensure the system is stable (i.e. that a set of steady-state transcript concentrations as per the above actually exists) the eigenvalues of A are investigated. If these all have strictly negative real parts, steady-state concentrations exist and the system is stable ([3]).

The result is a matrix X of transcript concentration deviations from unperturbed steady state, a true system connectivity matrix A, and a true perturbation matrix P. Corresponding estimates A_e, P_e are based on X using the algorithms tested, then compared to the true matrices to calculate recovery efficiency.

2.4.3 Cascade system

This network, simulations of which form the majority of this work, is a circular negative self-feedback cascade loop of ten genes, that is, each gene positively regulates the next with the last doubling back to the first. Also, each transcript negatively regulates its own transcription. Two additional genes with negative self-feedback only are also included to represent genes included in the experiment but with no regulatory connections to the main system. This is done to ensure that independent genes included in the experiment will not obstruct the analysis of any connected systems present. A graphical representation of the

connectivity matrix is shown in Figure 3, with the same matrix presented in detail in Table 1. Figure 2 shows the time progressions of the system for twelve different constitutive plasmid perturbations, one for each gene respectively, i.e., the step responses of the system.

The idea of this is to set up as simple a system as possible to function like a benchmark. Furthermore, the following strategy creates a very simple situation where only an algorithm that can distinguish direct and indirect responders may correctly retrieve the model. To obstruct reconstruction, an additional regulatory connection is added - one gene (1) has a variable positive regulatory influence on a non-adjacent gene (6) in the cascade. This is designed to display the simplest conditions under which the naïve expression-rate change method will be unable to correctly estimate the system parameters. The reason for this is that, with a large enough amplitude for this secondary regulatory motif, perturbation to gene 1 will increase the concentration of gene 6 even more, and from steady-state concentrations only, gene 6 will appear to be the most perturbed. Thus, estimation of regulation on gene 1 will be obstructed, and this difficulty will increase as the *amplification parameter* (that is, the element of the connectivity matrix which describes the effects of transcript 1 concentration on transcript 6 expression) does. Figure 1 shows the architecture of this system.

For analyzing the results of the algorithms on this system as the amplification factor increases, recovery scores are taken separately for gene 1 and the remaining genes. Comparing these scores yields a measure of how sensitive to this particular situation each method is, in effect, how well the method may separate primary and secondary perturbation targets.

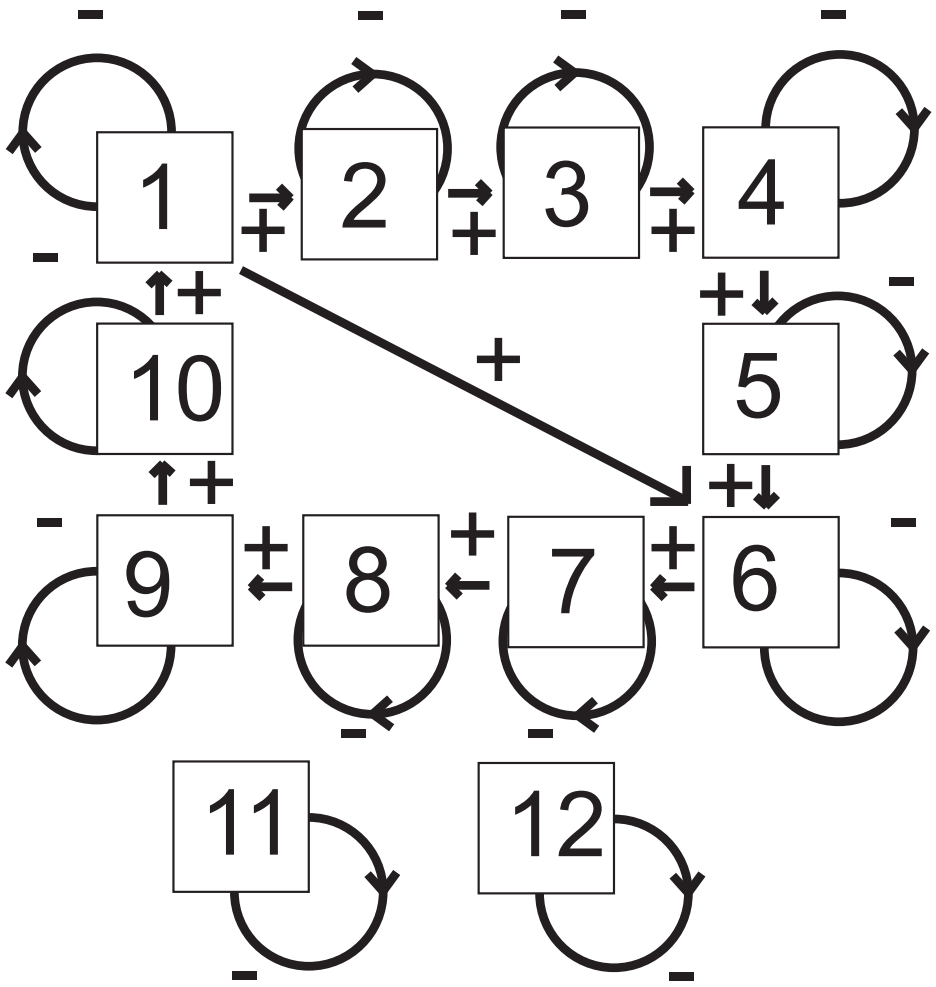

Figure 1: Network architecture of the circular cascade system with amplification factor = 0. Arrows represent connections, with signs corresponding to whether or not the influence increases or decreases expression.

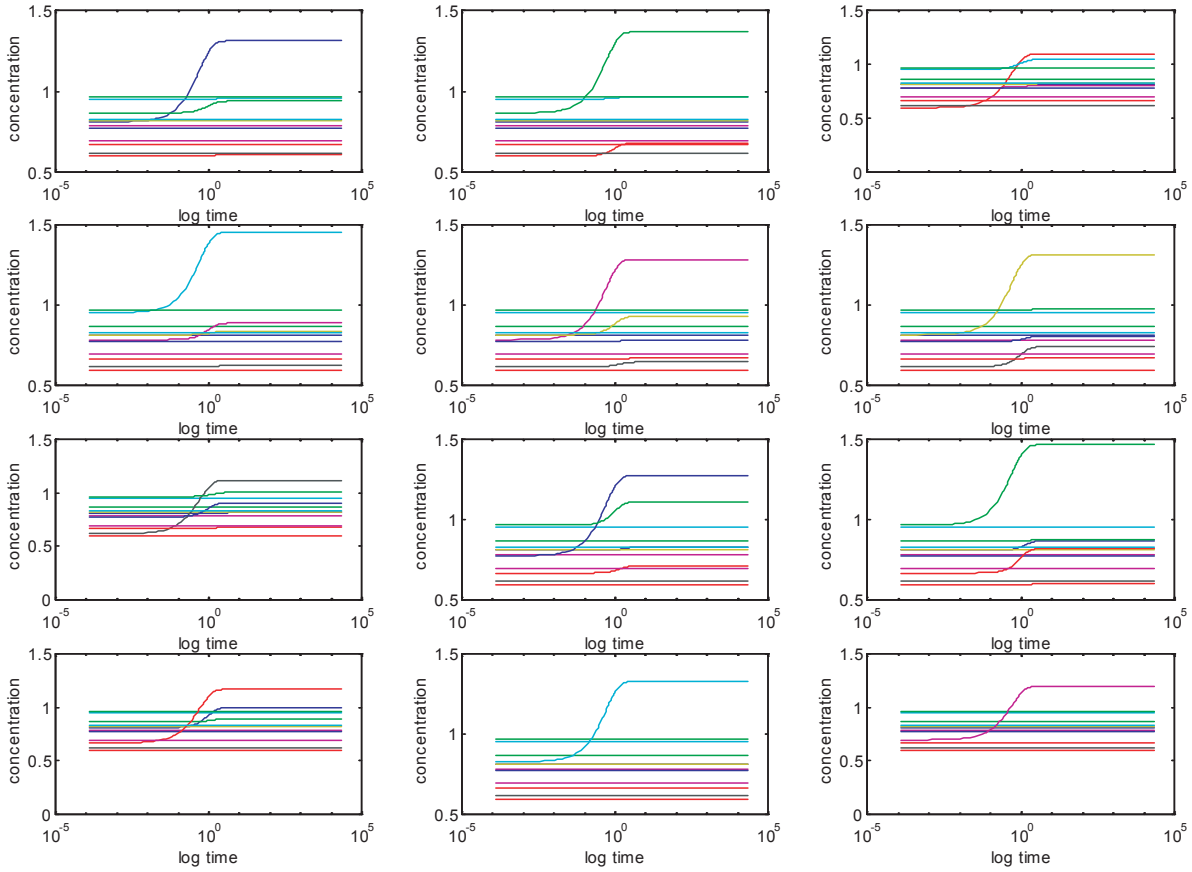

Figure 2: Response of circular cascade system presented in Figures 1 and 3 and Table 1 to perturbation of each gene $(1-12)$ in turn. Note that for most experiments, more than one gene changes expression, as a result of the system connectivity. Each graph represents a single possible perturbation experiment, in which one of the genes is given a constant overexpression from time $t = 0$ onwards. Thus, these are the step responses of the network. Graphs are ordered from left to right, row by row, in order of increasing gene index.

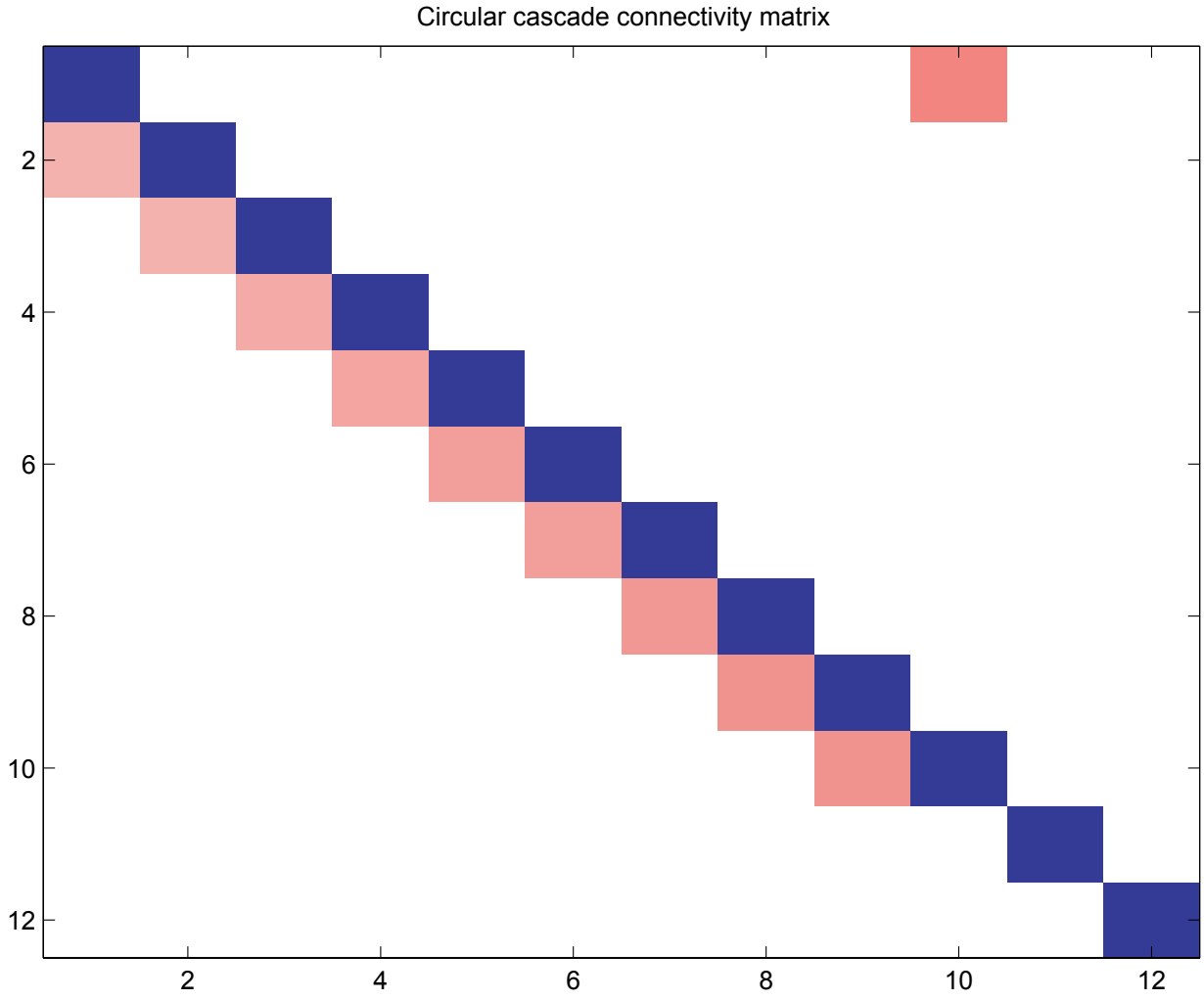

Figure 3: Circular cascade system connectivity (A) matrix. This is a graphical representation, in which red fields correspond to positive values, blue to negative.

-2.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.750	0.000	0.000
0.290	-2.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.330	-2.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.370	-2.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.410	-2.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.450	-2.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.490	-2.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.530	-2.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.570	-2.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.610	-2.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-2.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-2.000

Table 1: Circular cascade connectivity matrix (A) without cross-system connections (amplification factor = 0).

2.4.4 Full (Zak) system

To contrast with the previous system, a more complex simulation approach is also used. Presented by Zak et al ([28]), this system is pieced together from common regulatory motifs and with parameter values resembling biological reality reasonably well. It is quite complex and works using an expanded system containing promotor concentrations, transcripts, translation products and translation product dimers (hence the term "full" system). This system is included mainly to put the results from the linear simulations in context, to investigate whether the results hold for a more realistic system. The original implementation also employs a hybrid stochastic-deterministic model to simulate the effects of very low transcript concentrations, but this was not used in the current implementation as we focus on the effects of network architecture. Figure 4 shows the basic network architecture of the Zak system as used in this study.

When perturbing the Zak system, perturbations are made ten times as high as with the linear systems (in each time step while simulating a given experiment, a value of 1.0 is added to the transcript concentration of the gene being perturbed), to compensate for the different scale of this system. Figure 5 shows the dynamics of the Zak system when simulated from unperturbed steady state under perturbations to each gene in turn. As above, this was done using MATLABs built-in methods for stepwise integration.

-0.0599	-0.0010	-0.0000	-0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0001	-0.0220	-0.0000	-0.0000	0.0000	0.0932	0.0000	0.0000	0.0000	0.0000
0.0021	-0.0010	-0.0620	-0.0819	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
-0.0000	0.0000	-0.0000	-0.0620	0.0000	0.1009	0.0000	0.0000	0.0000	0.0000
0.0000	-0.0000	-0.0000	-0.0005	-0.0028	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	-0.0000	-0.0620	0.0000	0.0000	0.0000	0.0000
0.0000	-0.0000	-0.0041	0.0000	0.0000	-0.0000	-0.0620	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	-0.0016	-0.0010	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	-0.0194	0.2847
0.0000	-0.0000	-0.0041	0.0000	0.0000	-0.0000	0.0000	0.0000	0.0000	-0.0620

Table 2: Connectivity (A) matrix equivalent for the Zak system, as determined by the method in Section 2.4.5.

2.4.5 Estimating connectivity for the full system

Note that as the full system builds on complex relationships between many different molecular species to determine expression, there is no reason to expect transcript concentrations to conform to an equation of the form $AX = -P$ except as a Taylor expansion ([1]) around a given point in concentration space. However, as we are interested specifically in steady-state behaviour, we can linearize the system by Taylor expansion around its steady state, upon which the resulting connectivity matrix A and the linear equation system $A(Y - Y_0) = AX = -P$ will describe the system when the transcript concentrations Y are close to the steady state Y_0. In this region, then, it is possible to compute such a connectivity matrix equivalent, then compute connectivity matrices via MNI and NIR and compare these with the connectivity matrix equivalent to determine algorithm performance on the full system close to a steady state.

To compute this connectivity matrix equivalent around a steady state, we use the following approach, as described in ([10]). We begin by stating the Implicit Function theorem.

Assuming that the following M identities exist.

$$\begin{cases} F_1(x_1, x_2, ..., x_N, z_1, ..., z_M) &= 0 \\ F_2(x_1, x_2, ..., x_N, z_1, ..., z_M) &= 0 \\ ... & ... \\ F_M(x_1, x_2, ..., x_N, z_1, ..., z_M) &= 0 \end{cases} \tag{15}$$

Let J be an $M \times M$ matrix for which it holds that

$$J = \begin{pmatrix} \frac{\partial F_1}{\partial z_1} & \frac{\partial F_1}{\partial z_2} & ... & \frac{\partial F_1}{\partial z_M} \\ \frac{\partial F_2}{\partial z_1} & \frac{\partial F_2}{\partial z_2} & ... & \frac{\partial F_2}{\partial z_M} \\ ... & ... & ... & ... \\ \frac{\partial F_M}{\partial z_1} & \frac{\partial F_M}{\partial z_2} & ... & \frac{\partial F_M}{\partial z_M} \end{pmatrix} \tag{16}$$

Furthermore, let J be nonsingular at the point $(\mathbf{x}_o, \mathbf{z}_o)$. By the Implicit Function Theorem, the system in Equation 15 defines the following set of locally valid unique functions

$$z_n = g_n(x_1, x_2, ..., x_N) \qquad n = 1, 2, ..., M. \tag{17}$$

which can be derived implicitly as

$$D_{\mathbf{x}}\mathbf{F} = \frac{\partial \mathbf{F}}{\partial \mathbf{x}} + \frac{\partial \mathbf{F}}{\partial \mathbf{z}}(\mathbf{x}, \mathbf{g}(\mathbf{x}))\frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} = 0 \tag{18}$$

Consider the network dynamical model

$$\frac{d\mathbf{x}_m}{dt} = \mathbf{f}_m(\mathbf{x}_m, \mathbf{x}_p, \mathbf{x}_{pm}) \tag{19}$$

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{f}_p(\mathbf{x}_m, \mathbf{x}_p, \mathbf{x}_{pm}) \tag{20}$$

$$\frac{d\mathbf{x}_{pm}}{dt} = \mathbf{f}_{pm}(\mathbf{x}_m, \mathbf{x}_p, \mathbf{x}_{pm}) \tag{21}$$

for mRNA concentrations $\mathbf{x}_m$, protein concentrations $\mathbf{x}_p$, and promoter concentrations $\mathbf{x}_{pm}$. Further, we assume that the system is in steady state, and hence

$$0 = \mathbf{f}_p(\mathbf{x}_m, \mathbf{x}_p, \mathbf{x}_{pm}) \tag{22}$$

$$0 = \mathbf{f}_{pm}(\mathbf{x}_m, \mathbf{x}_p, \mathbf{x}_{pm}) \tag{23}$$

Let $\mathbf{F} = [\mathbf{f}_p \quad \mathbf{f}_{pm}]^T$. For N_p protein equations and N_{pm} promoter equations, we assume that the $(N_p + N_{pm}) \times (N_p + N_{pm})$ matrix J is non-singular, then by the Implicit Function Theorem, this means that Equations 22 and 23 implicitly defines the N_p locally unique functions $\mathbf{x}_p = \xi_p(\mathbf{x}_m)$ and the N_{pm} locally unique functions $\mathbf{x}_{pm} = \xi_{pm}(\mathbf{x}_m)$. By making use of implicit differentiation $D_{\mathbf{x}_m}\mathbf{F} = 0$, we may then acquire their derivatives.

Hence, for small perturbations of $\mathbf{x}_m$ from its steady state value, implicitly well defined perturbed values of the protein and promoter concentrations exist, which can be determined from the vector valued functions ξ_p and ξ_{pm}.

We then define

$$\tilde{\mathbf{f}}_m(\mathbf{x}_m) = \mathbf{f}_m(\mathbf{x}_m, \xi_p(\mathbf{x}_m), \xi_{pm}(\mathbf{x}_m)) \tag{24}$$

and may then describe the local mRNA dynamics around the steady state as

$$\frac{d\mathbf{x}_m}{dt} = \tilde{\mathbf{f}}_m(\mathbf{x}_m). \tag{25}$$

By Taylor series expansion ([10]) of these functions around the steady state, we have

$$\frac{d\mathbf{x}_m}{dt} = A(\mathbf{x}_m - \mathbf{x}_m^o). \tag{26}$$

for $A = \frac{\partial \tilde{\mathbf{f}}_m}{\partial \mathbf{x}_m}|_{\mathbf{x}_m = \mathbf{x}_m^o} = \frac{\partial \mathbf{f}_m}{\partial \mathbf{x}_m} + \frac{\partial \mathbf{f}_m}{\partial \mathbf{x}_p}\frac{\partial \xi_p}{\partial \mathbf{x}_m} + \frac{\partial \mathbf{f}_m}{\partial \mathbf{x}_{pm}}\frac{\partial \xi_{pm}}{\partial \mathbf{x}_m}|_{\mathbf{x}_m = \mathbf{x}_m^o}$.

Define $\mathbf{z}_m(t) = \mathbf{x}_m(t) - \mathbf{x}_m^o$, then apply, as in a perturbation experiment, an external constitutive perturbation $\mathbf{u}$ on mRNA transcription. Local dynamics can then be linearly approximated as

$$\frac{d\mathbf{z}_m}{dt} = A\mathbf{z}_m + \mathbf{u}. \tag{27}$$

which yields an equivalent of the connectivity matrix A around a steady state even for a system where no such matrix is explicitly defined. By applying this algorithm to the simulated system, where derivatives may be computed numerically using difference quotients at any given point, this connectivity matrix equivalent is computed and used as the closest approximation of the "true" system for evaluating perturbation method performance on the full system. Figure 6 shows a graphical representation of this matrix, which is presented in Table 2.

2.4.6 Random architecture system model

Another simulated model uses a random architecture, vaguely similar to the Zak system ([28]) in type, for each experiment. This is done to see how well results from the cascade system holds up in simulations of more complex linear models.

The random architecture model assumes the following: the system includes between 10 and 12 genes. Of these, 2 to 5 are "hubs", which influence several other genes in either positive or negative direction. The rest form cascades moving off from the hubs. Parameter values are on the same scale as for the circular cascade system above.

Figure 7 shows an example architecture obtained this way, Figure 8 a graphical representation of its connectivity matrix, which is presented in Table 3, and Figure 9 show the result of perturbing this system from unperturbed steady state one gene at a time. As above, these results were obtained by using MATLABs built-in methods for ode solving.

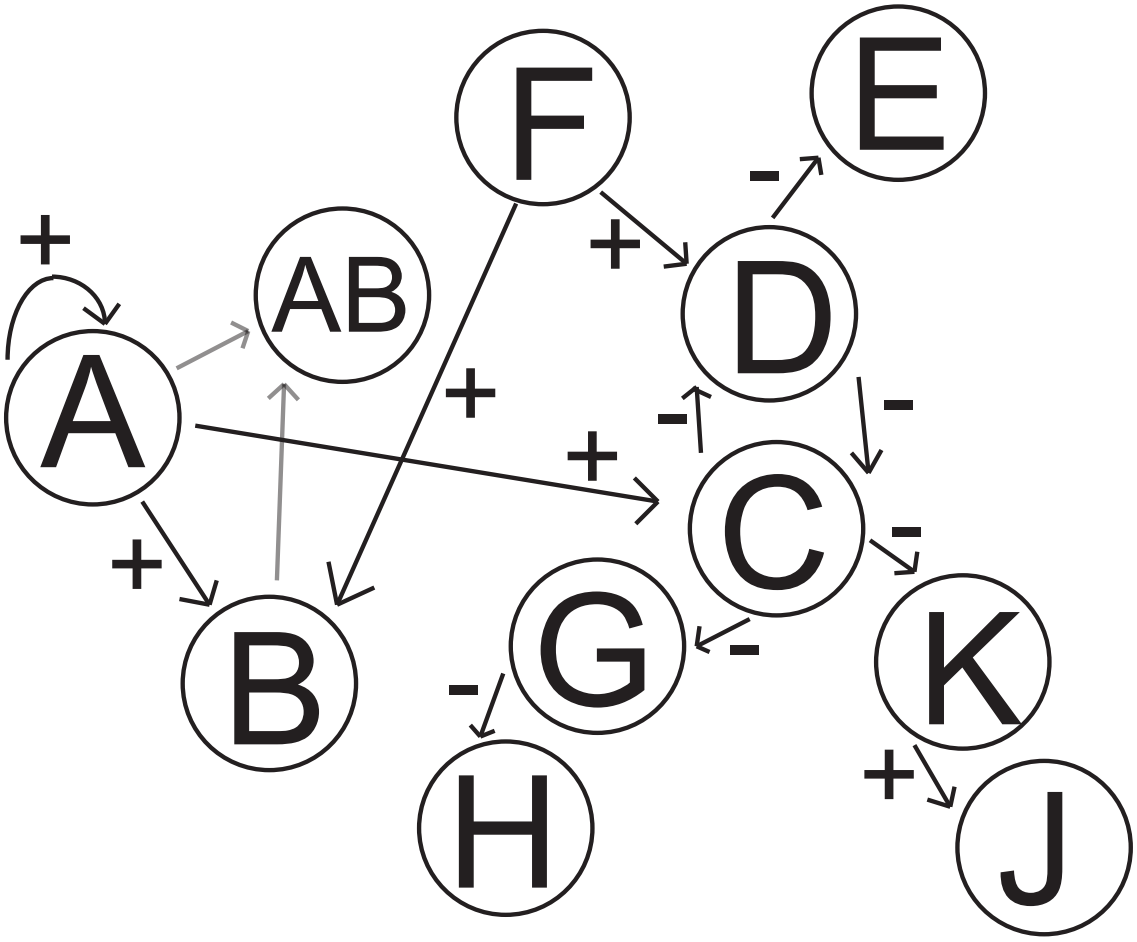

Figure 4: Network architecture of the Zak system. Arrows represent connections, with signs corresponding to whether or not the influence increases or decreases expression. The gray arrows represent sequestering of gene products A and B into the heterodimer form AB.

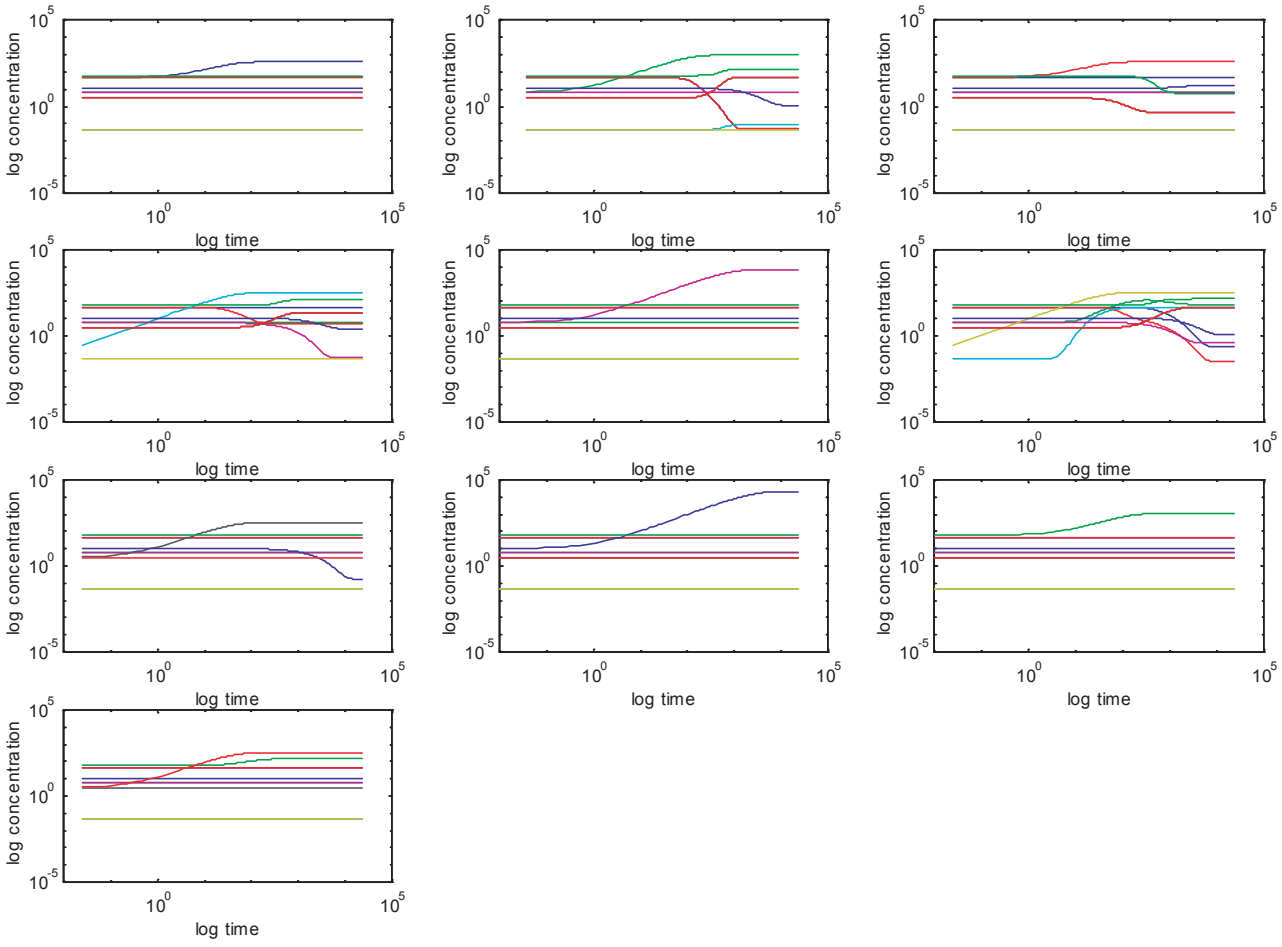

Figure 5: Response of the Zak system to perturbation of each gene (A, B, C, D, E, F, G, H, J, K) in turn. Note that for most experiments, more than one gene changes expression, as a result of the system connectivity. Each graph represents a single possible perturbation experiment, in which one of the genes is given a constant overexpression from time $t = 0$ onwards. Thus, these are the step responses of the network. Graphs are ordered from left to right, row by row, in order of increasing gene index.

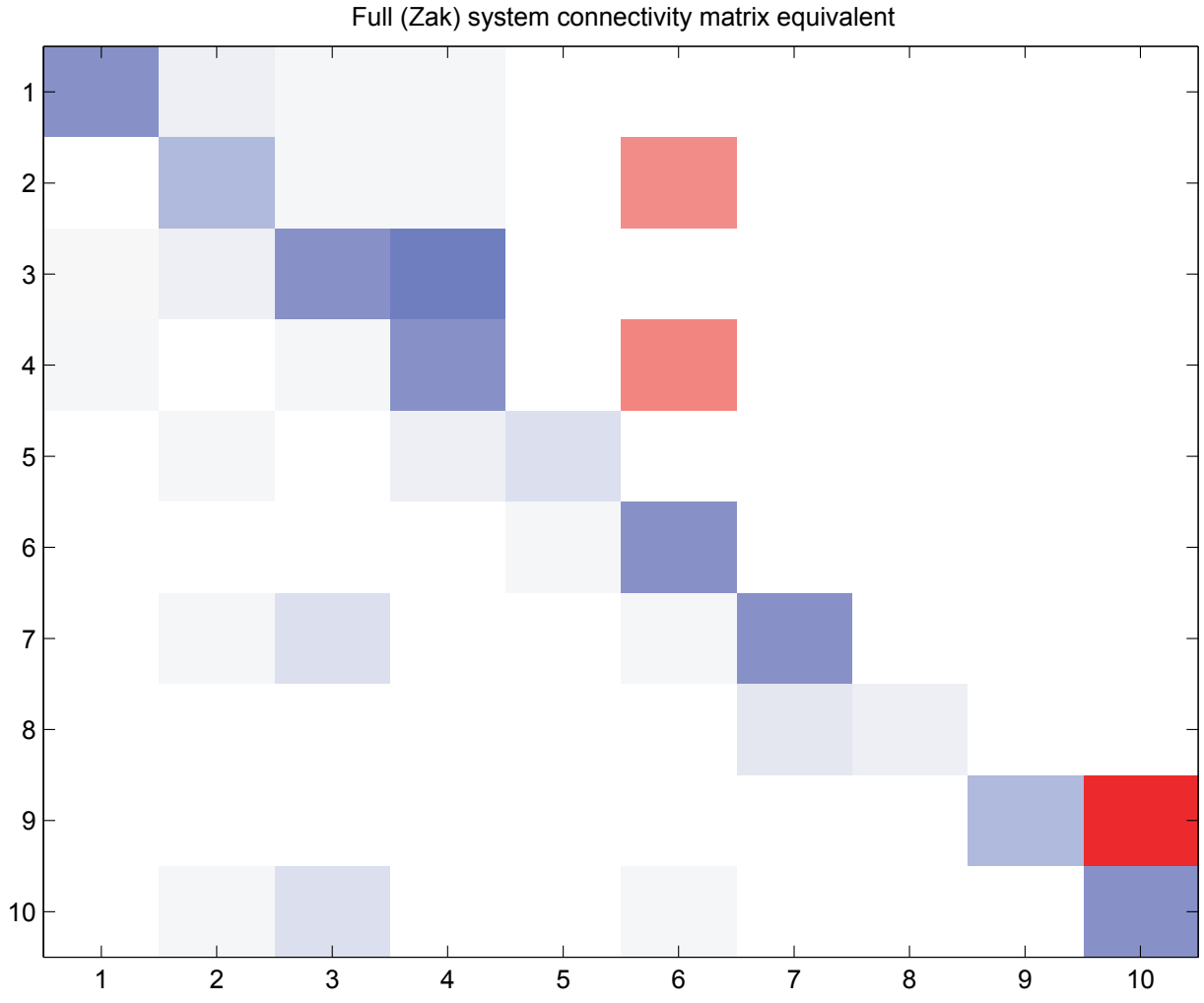

Figure 6: Connectivity (A) matrix equivalent for the Zak system, as determined by the method in Section 2.4.5. This is a graphical representation, in which red fields correspond to positive values, blue to negative.

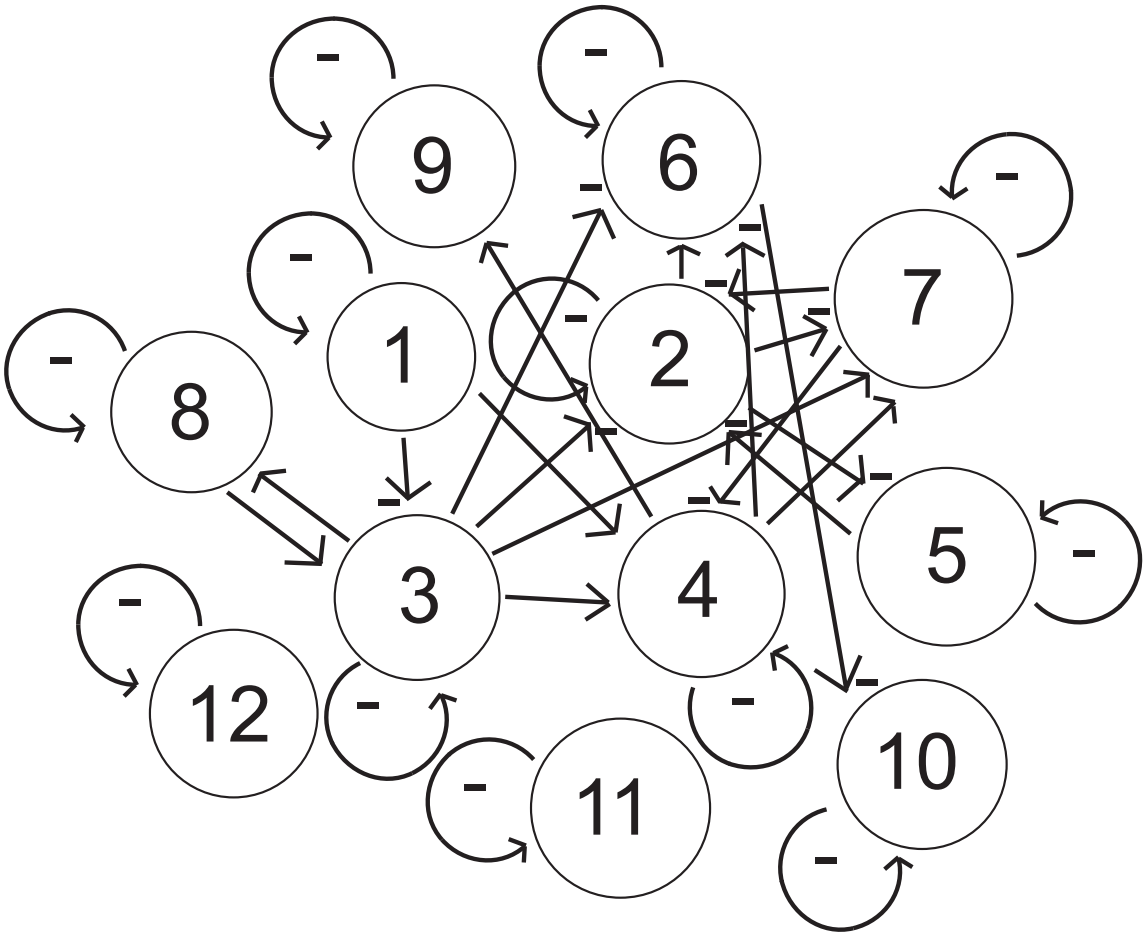

Figure 7: Sample random architecture system connectivity. Arrows represent connections, with signs corresponding to whether or not the influence increases or decreases expression.

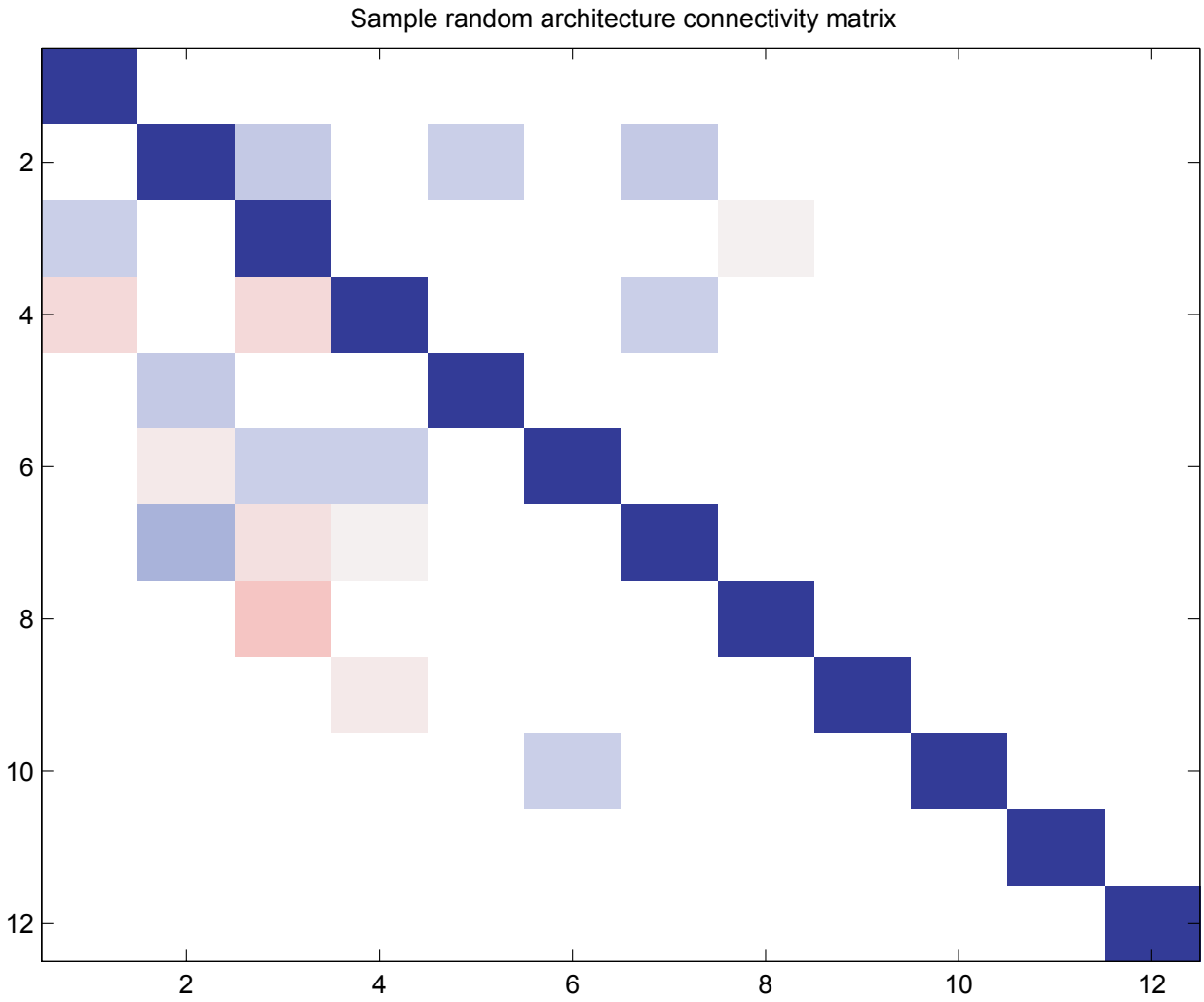

Figure 8: Sample random architecture system connectivity. Red fields correspond to positive values of the matrix, blue to negative.

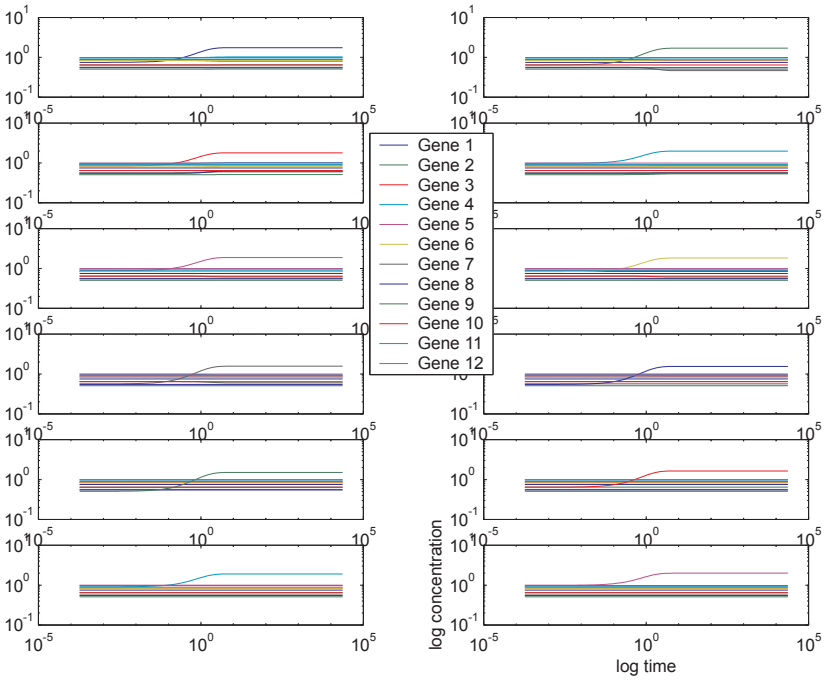

Figure 9: Response of the sample random architecture network of Figure 8 and Table 3 to perturbation of each gene $(1-12)$ in turn. While in most experiments more than one gene actually reacts to the perturbation, because of the system connectivity, this is hard to see in the graph as the connections in this particular example are relatively weak. Each graph represents a single possible perturbation experiment, in which one of the genes is given a constant overexpression from time $t = 0$ onwards. Thus, these are the step responses of the network. Graphs are ordered from left to right, row by row, in order of increasing gene index.

-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	-0.952	-0.040	0.000	-0.027	0.000	-0.042	0.000	0.000	0.000	0.000	0.000
-0.034	0.000	-1.045	0.000	0.000	0.000	0.000	0.013	0.000	0.000	0.000	0.000
0.044	0.000	0.037	-1.000	0.000	0.000	-0.027	0.000	0.000	0.000	0.000	0.000
0.000	-0.040	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.021	-0.026	-0.029	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	-0.097	0.034	0.011	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.087	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.024	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	-0.034	0.000	0.000	0.000	-1.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000

Table 3: Sample random architecture connectivity matrix (A). This matrix is the same as that visualized in Figures 7, and 8 and the perturbation response of which is shown in Figure 9.

2.5 Perturbation sets

2.5.1 Single plasmid-type perturbations

The simplest form of experiment for the above systems involves perturbing each gene exactly once, in as many experiments as there are genes. The perturbation matrix will be some multiple of the identity matrix, for these simulations simply the identity matrix (i.e. the amplitude of each perturbation is 1.0). This would involve making a specific plasmid perturbation to every gene in a set of interest, perhaps those found to be differentially expressed under some condition or in some cell type. By including all genes once in this perturbation set we ensure that, for the purposes of this work, the space of genetic dynamics is spanned by the perturbations. In the experiments performed here, columns of the perturbation matrix are randomly permuted to avoid possible method artifacts, such as a method incorrectly favouring the identity matrix.

2.5.2 Pairwise plasmid-type perturbations

For comparison, another possibility is perturbation of every set of two genes. This would reflect more complex external influences as well as more experiments in total. In this study, what is done is simulating the system using the single perturbation set above, then trying to recover the network model from the results.

Then, the system is simulated using every set of perturbation of two different genes simultaneously, and the ability of the previously taken connectivity matrix to recover these perturbations is investigated. This is done in order to ensure that the methods can operate even when test set and training set do not overlap. Obviously, the pairwise plasmid-type perturbation set represents far more experiments than would ever be performed in a real example, and is used here to investigate method performance under ideal circumstances. Just as for the single plasmid-type permutation set, columns are randomly permuted for each run.

2.6 Experiments

2.6.1 Experiment setup

The basic setup of the experiments performed is as follows: for a given simulated system from those described above, apply either single or pairwise perturbations and simulate to steady-states. Then separate the experiments into training and test sets; in the case of single plasmid perturbations, we are interested primarily in the application of the method for charting regulatory networks and thus estimate the connectivity from the training set, then use this estimate to estimate the external perturbations used to generate the training set. In the case of pairwise plasmid perturbations, we use a training set of single-gene perturbation experiments to estimate connectivity, then use this estimate to try to recover the external perturbations used to generate a test set, which was generated using every combination of pairwise gene perturbations. By doing this, we seek to ensure that the method does not require the test set experiments to be present in the training set.

In each experiment, some parameter (such as measurement noise), which we call the *obstructing factor*, is varied which is expected to make system recovery more difficult. As this parameter increases, decrease in performance is monitored both for recovery of connectivity and recovery of external perturbations, across the various methods described above. The purpose of this is to allow comparisons of the methods under a variety of obstructing values of parameters.

For every value of the parameter which is varied, twenty simulations are made, each using random unperturbed steady state values (evenly distributed between

0.5 and 1.0). These are analyzed independently and mean and variance for the results are taken.

To represent common laboratory practices as well as estimate variance of measurements, all simulations are made in triplicate, and then averaged. Data is also preprocessed as per the ([7]). In test runs, it was found that for these datasets, use of z-score statistics (that take into account variation in the data) made little difference for the results, and hence, results using these are omitted.

2.6.2 Dependence on initial guess

Several variants of MNI are used, with different initial guess at the connectivity matrix prior to the first iteration of the algorithm. These are in effect treated as different methods, and as such, the comparison is made in each of the following experiment setups. Details of these variants (listed in the plots as *MNI - alternate start guess 1* and *MNI - alternate start guess 2*) are described in section 2.2.4 above.

2.6.3 Other MNI variants

The previously described variants using either fixed 50% sparsity for connectivity and perturbation matrices (listed in the plots as *MNI - enforce* 50% *sparse A/P*) or assumes at most one experiment perturbs each gene (listed in the plots as *MNI - enforce sparse P*) are likewise applied to the datasets. See section 2.2.4 for details.

2.6.4 Dependence on noise

To investigate the effect of stochastic measurement noise on system recovery, different degrees of noise was used. This is stochastic (Gaussian) noise, where for each measurement, a percentage of the average transcript concentration of the dataset is multiplied with a normally distributed pseudo-random number between -1.0 and 1 and added to the measurement to simulate measurement errors of all kinds. The analysis is made for the above circular cascade system,

with the amplification factor set to 0 (that is, for a pure circular cascade) using both single and pairwise perturbation sets, as well as for the random architecture and full systems using single perturbations. This not only yields a comparison of the effects of increasing noise on the different methods, but also for the different types of system. The noise percentage is varied between 0% and 400% of the average transcript concentration.

2.6.5 Dependence on amplification factor

For the linear circular cascade system, system recovery should become more difficult using expression changes only as the transcriptional influence of gene 1 on gene 6 becomes stronger, just as described in section 2.4.3. By varying the corresponding element in the generating connectivity matrix (that is, by increasing the amplification factor) from zero upwards (while remaining within system stability), the various methods can be compared with respect to this effect. The core issue is whether the methods tested for recovering the perturbation matrix can outperform the alternative of just looking at expression level change for the particularly obstructed gene A.

2.6.6 Dependence on MNI σ factor

The MNI algorithm contains a sparsity assumption which is regulated by a parameter σ. In the original MNI article it is stated that simulations have shown a value of $\sigma = 0.25$ as suitable, but this is not described or motivated in any detail. To clarify the effects of varying the σ parameter, σ is allowed to vary between 0.0 and 1.0. For each value of the parameter, 20 datasets are simulated for each of the three networks described above, without any measurement noise or other obstructing factors, and MNI is applied to the dataset using that value for the σ parameter.

2.6.7 Result corroboration MNI - NIR

Finally, the present implementations of the MNI and NIR methods were applied to the dataset used in the original NIR article ([9]). The recovered matrices are compared to those actually used as well as the results from the NIR method.

This data was acquired by Gardner and associates by growing *E. coli* cells with controllable plasmids added as per the requirements of the NIR method, then measuring both transcript concentrations in steady state for the nine genes of the SOS pathway and the magnitude of the plasmid perturbation.

3 Results

3.1 Overview

This section (3) begins with an overview of its contents (3.1), then proceeds (3.2) by describing the methods for evaluating the perturbation data analysis methods on simulated data (3.2.1) as well as describing how the results are presented (3.2.2). Following this, the actual results are shown, experiments for testing method dependence on measurement noise (3.3) and their interpretation (3.3.1), experiments for testing method capacity to distinguish direct from indirect responders (3.4) and their interpretation (3.4.1), experiments for testing the dependence of MNI on its internal σ sparsity parameter and their interpretation (3.5.1), and last, application of MNI to the dataset on which NIR was originally tested (3.6) and an interpretation (3.6.1) of these results.

3.2 Evaluating results

3.2.1 Matrix comparison

There are several ways in which methods considered in this thesis may be evaluated. All rely on applying them to known networks, then comparing estimated results to the true nets.

To evaluate network reconstruction, one may either look at the connectivity matrix A itself, or see to which degree the estimated interdependency matrix may be used to retrieve the actual perturbed steady states from the associated perturbation matrix. The former is interesting in that it reflects how well the algorithm can recover system characteristics in themselves, and so indicate its usefulness as a tool to study genetic regulation for its own sake. The latter is interesting as it shows the algorithms usefulness in molecular mechanism detection and similar applications. Performance in these two respects is connected; perfect recovery of the A matrix would entain perfect recovery of the P matrix. However, a method could conceivably be better at predicting external influences than regulatory connections.

Both the perturbation ranking matrix and the connectivity matrix are, at least for MNI, potentially differentially scaled row-wise ([7]). That is to say, the MNI output is a connectivity matrix which in the ideal case differs from the actual matrix A only by multiplying each row by some constant. Thus, it is not a priori certain that the element size-order in a column reflects the underlying biology. The comparisons between estimate and reality are thus made on the basis of ranking within rows. The highest elements in a given row are likely to represent actual connections or influences.

When comparing true and estimated perturbation and connectivity matrices, the following approach has been used. Let D_t be the true $N \times M$ matrix, which we assume to be relatively sparse. Let D be the matrix computed by the algorithm we test. Let d_i, d_{ti} be the ith row of these matrices. Let n_i be the number of nonzero elements in d_{ti}. Let u_i be a vector with the elements of d_i ordered by absolute value, and define v_i so that $u_i(j) = d_i(v_i(j))$ for $j = 1..M$ - that is, v_i are the size-ordered indices for the ith row of the computed matrix. Define $\delta(a, b)$ as 1 if a, b are both nonzero and have the same sign, 0 otherwise. Then, our performance metric e, or the *average recovery score*, is defined as

$$e = \sum_{i=1..N} \frac{\sum_{j=1..n_i} \delta(u_i(j), d_{ti}(v_i(j)))}{n_i} \tag{28}$$

That is to say, e is the proportion of elements in matrix D that correspond to actual influences or connections in matrix D_t that are neither outranked by other elements in matrix D nor assigned the wrong sign. To understand the rationale behind this metric, consider a row of D_t. This row will contain a small number of nonzero elements. As we seek to capture the existence and type of the connections or perturbations represented in D_t, rather than their size, we have no way of ranking these internally, but they can be singled out as those indices of this row that correspond to actual connections or perturbations. For the corresponding row of the recovered matrix D, we note that the elements of that row can be ranked according to absolute value. If the elements of that row that rank highest are the same elements as the nonzero elements in this row of D_t, and also have the same signs as these, then the highest-ranking elements of this row of D captures the connections or perturbations represented by the corresponding row of D_t well. The metric e measures the degree to which this similarity of representation exists.

This metric ranks from 0.0 to 1.0, for complete recovery of matrix characteristics. This means that type of regulation/perturbation is considered (in the form of its sign), but not relative size of such influences. This recovery metric is used in all method trials. Obviously, this approach only makes sense when the true system matrices are sparse. In these experiments, it is sometimes of interest to compare performance between genes or sets of genes. In this case, the metric is simply taken separately over the corresponding submatrices, that is, for the two sets of genes I_1 and I_2 we have

$$e_1 = \sum_{i \in I_1} \frac{\sum_{j=1..n_i} \delta(u_i(j), d_t i(v_i(j)))}{n_i}, e_2 = \sum_{i \in I_2} \frac{\sum_{j=1..n_i} \delta(u_i(j), d_t i(v_i(j)))}{n_i} \tag{29}$$

In particular, this is done to separate the performance on one particular gene out from the rest.

3.2.2 Result presentation

Almost all of the experiments performed yield results of a similar nature - for each method there is a series of average recovery scores as the obstructing factor increases. In the experiments performed using the circular cascade system with increasing amplification factor, separate such scores exist for gene 1 and the remaining genes, and for all experiments, separate scores are taken for connectivity and perturbation recovery. Most graphs have logarithmic x-axes as the obstructing parameter has been taken over a logarithmic range.

To display these results, scores are plotted as a function of the obstructing factor. To interpret these graphs properly, note the comparison on one hand between methods, including the benchmark of the subtraction method, on the other hand, for the amplification factor experiments, between gene 1 (which becomes harder to recover parameters for as the amplification factor increases) and the other genes.

For comparing the results from the original NIR paper ([9]) with the results from MNI applied to the same dataset, the recovered matrices are presented but also graphically displayed as cell graphs.

3.3 Dependence on measurement noise

Four main experiments were made testing how increasing measurement noise affected recovery. The circular cascade system was perturbed using both the single and the pairwise perturbation sets above. The full system and the random architecture system was perturbed using single perturbations only. For each of these, the following methods were applied to recover the connectivity matrix: MNI, the two alternate start guesses, MNI with enforcing P sparsity and 50% A/P sparsity, respectively (see sections 2.2.4 and 2.6 for details on these variants), and NIR. For perturbation matrix recovery, using the expression level change data is also applied. Figures 10, 11, 12 and 13 show the results of each of these experiments.

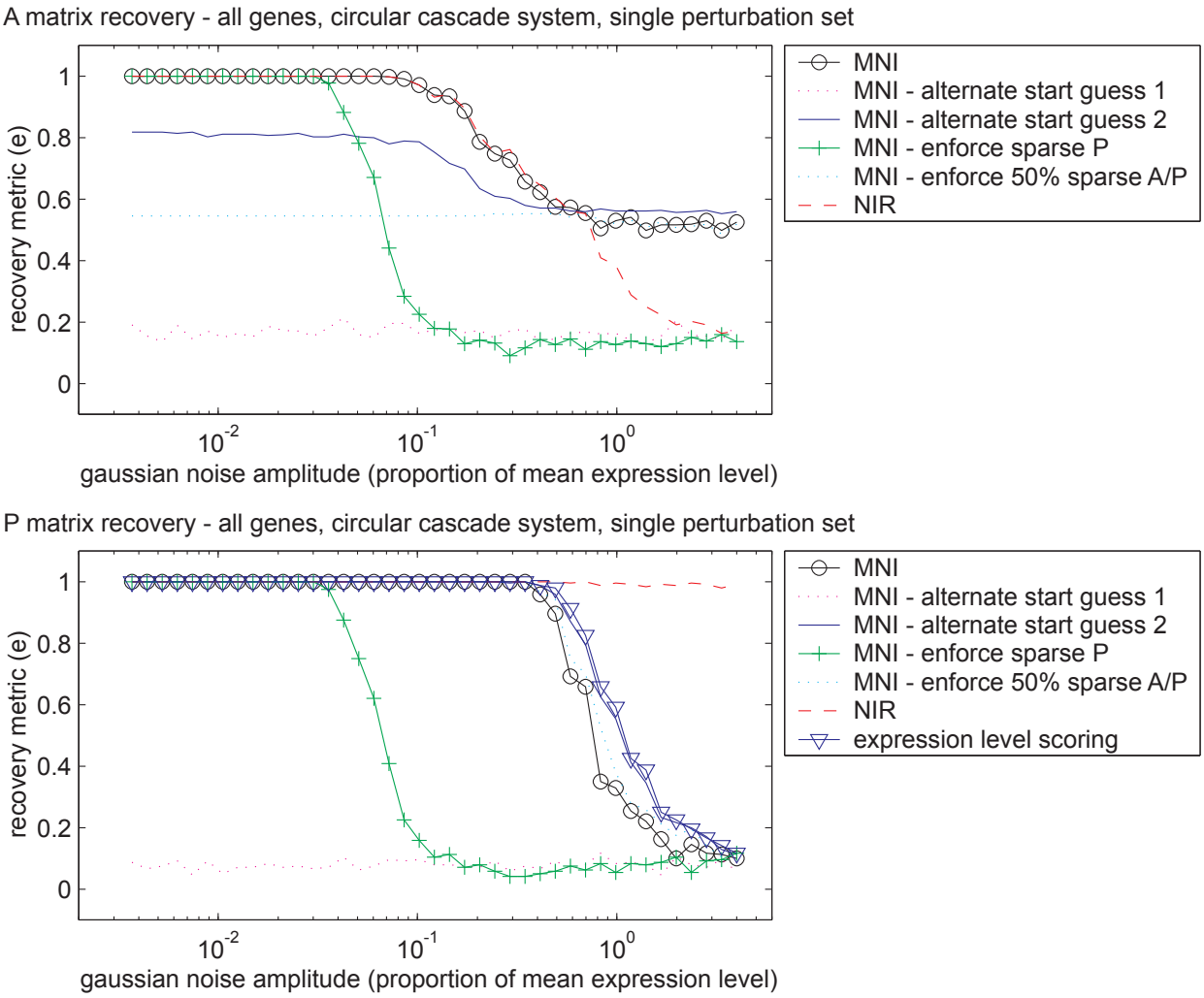

Figure 10: Matrix recovery under increasing measurement noise, single perturbations, circular cascade system. Each graph represents a method or method variant, each point the average recovery metric (e) score over 20 simulated datasets generated using that level of measurement noise.

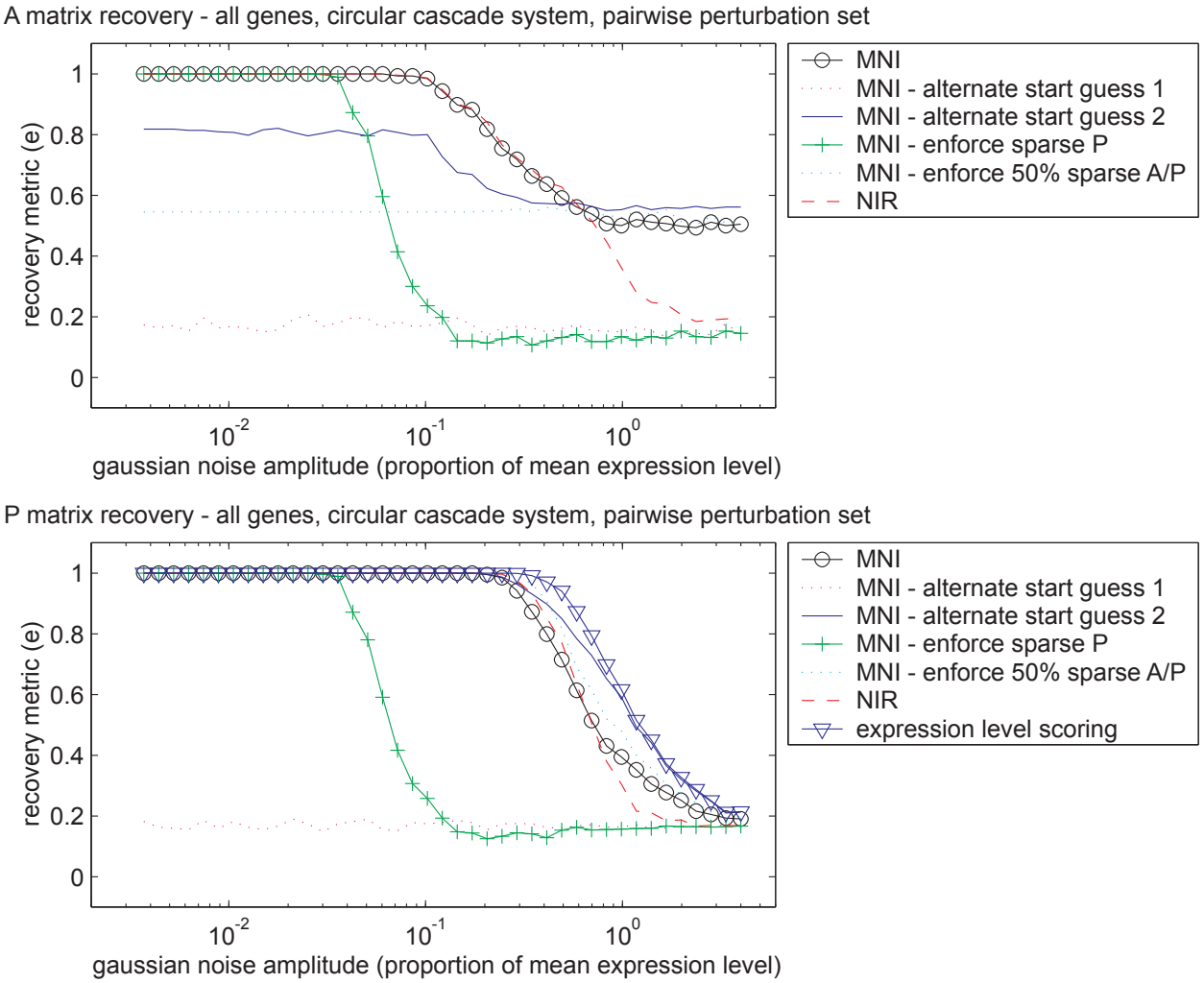

Figure 11: Matrix recovery under increasing measurement noise, pairwise perturbations, circular cascade system. Each graph represents a method or method variant, each point the average recovery metric (e) score over 20 simulated datasets generated using that level of measurement noise.

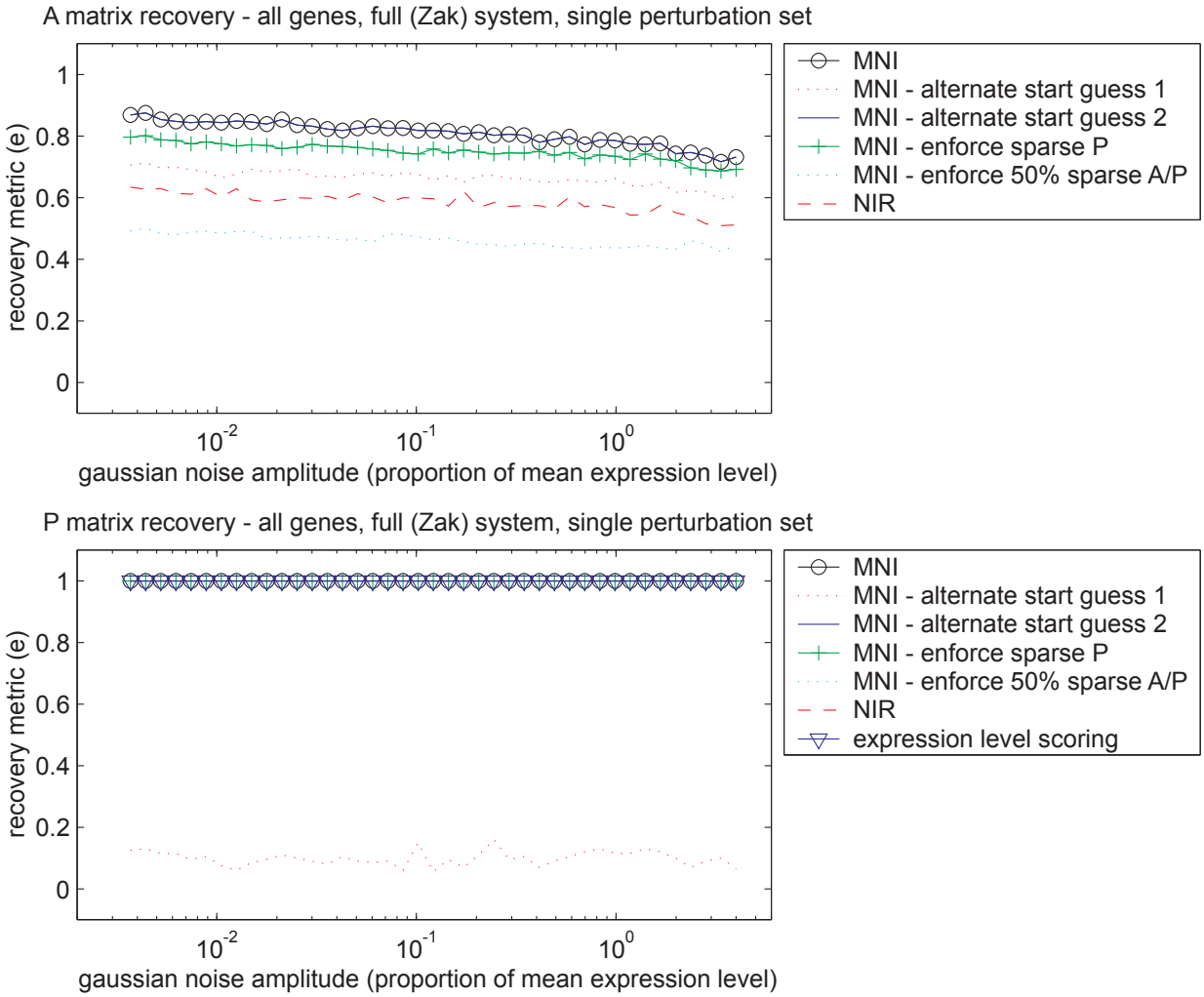

Figure 12: Matrix recovery under increasing measurement noise, single perturbations, full (Zak) system. Each graph represents a method or method variant, each point the average recovery metric (e) score over 20 simulated datasets generated using that level of measurement noise.

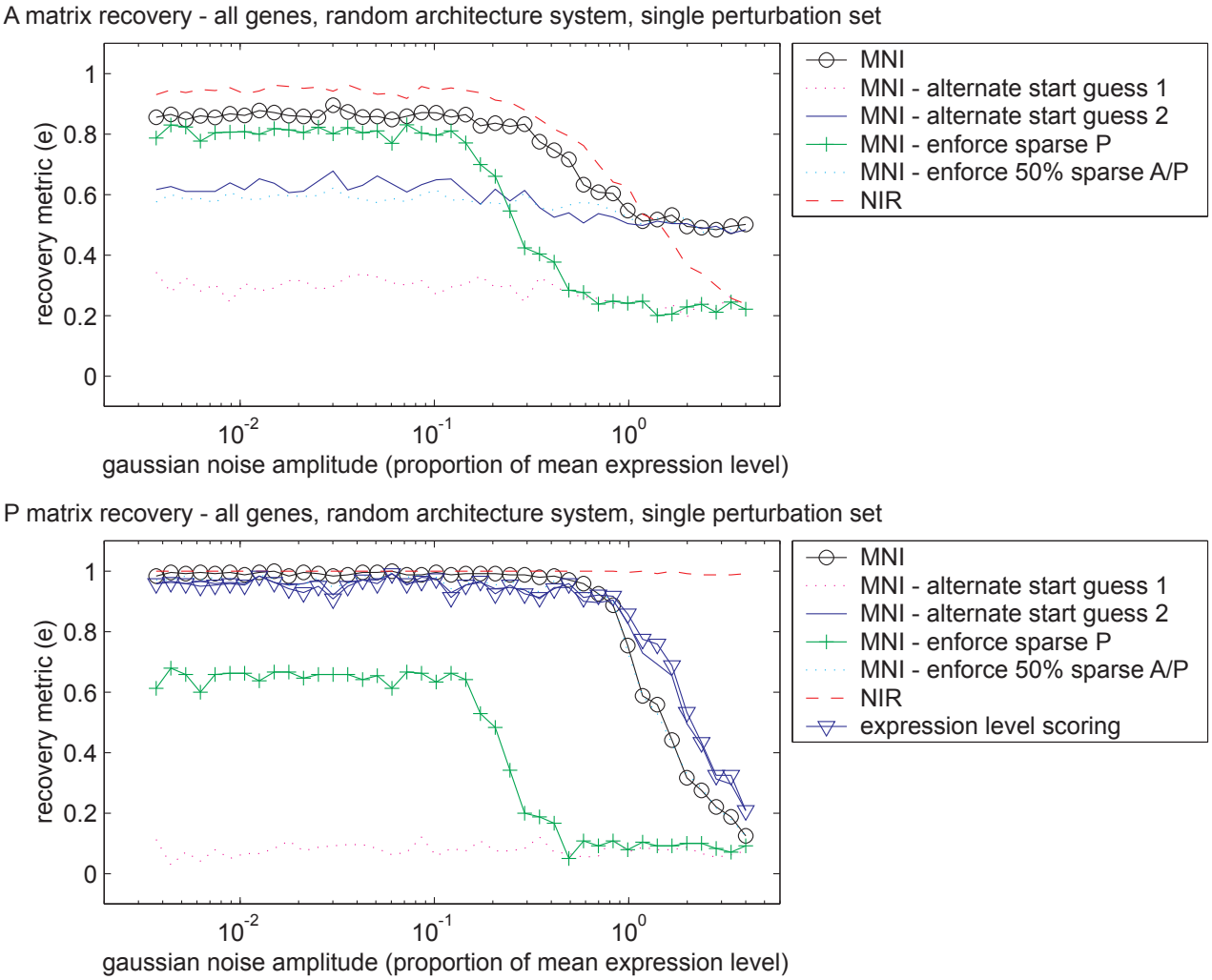

Figure 13: Matrix recovery under increasing measurement noise, single perturbations, random architecture. Each graph represents a method or method variant, each point the average recovery metric (e) score over 20 simulated datasets generated using that level of measurement noise.

3.3.1 Interpretation

There are some core features of the noise experiment results. First of all, both MNI and NIR succeed in recovering the model matrices satisfactorily up to a certain degree of noise (most of the time around a gaussian noise amplitude of 50% of the average expression level). When recovery efficiency does drop, it does not do so gradually, but rather relatively abruptly. This implies that the methods locate and amplify a signal where it is present, but as the effects of noise rise above those of signal, no recovery is possible. NIR is not significantly better than MNI at recovering the connectivity (A) matrices. It does perform better for the perturbation (P) matrices, but as it has access to the true solution in these cases (test set is in training set) this really means very little. None of the suggested variant methods perform better than either MNI or NIR, nor does MNI perform significantly better than expression level analysis in these experiments. It can further be seen that performance drops quicker in respons to noise for the more complex network types. Moreover, NIR performs relatively poorer for the more complex Zak system. This appears to be because cross-validation under these circumstances selects a connectivity which is too low to represent the system dynamics with full accuracy.

3.4 Dependence on amplification factor

For the circular cascade system, experiments were made using both single and pairwise perturbations, increasing the strength of the cross-system connection parameter (the 'amplification factor'). Results are shown separately for gene 1 and the remaining genes, as we seek to investigate how well the methods work under the particular sets of circumstances applied to gene 1. For single perturbations, Figure 15 shows recovery of gene 1 whereas Figure 14 shows average connectivity and perturbation recovery for the remaining genes. For pairwise perturbations, Figures 17 and 16 show corresponding results.

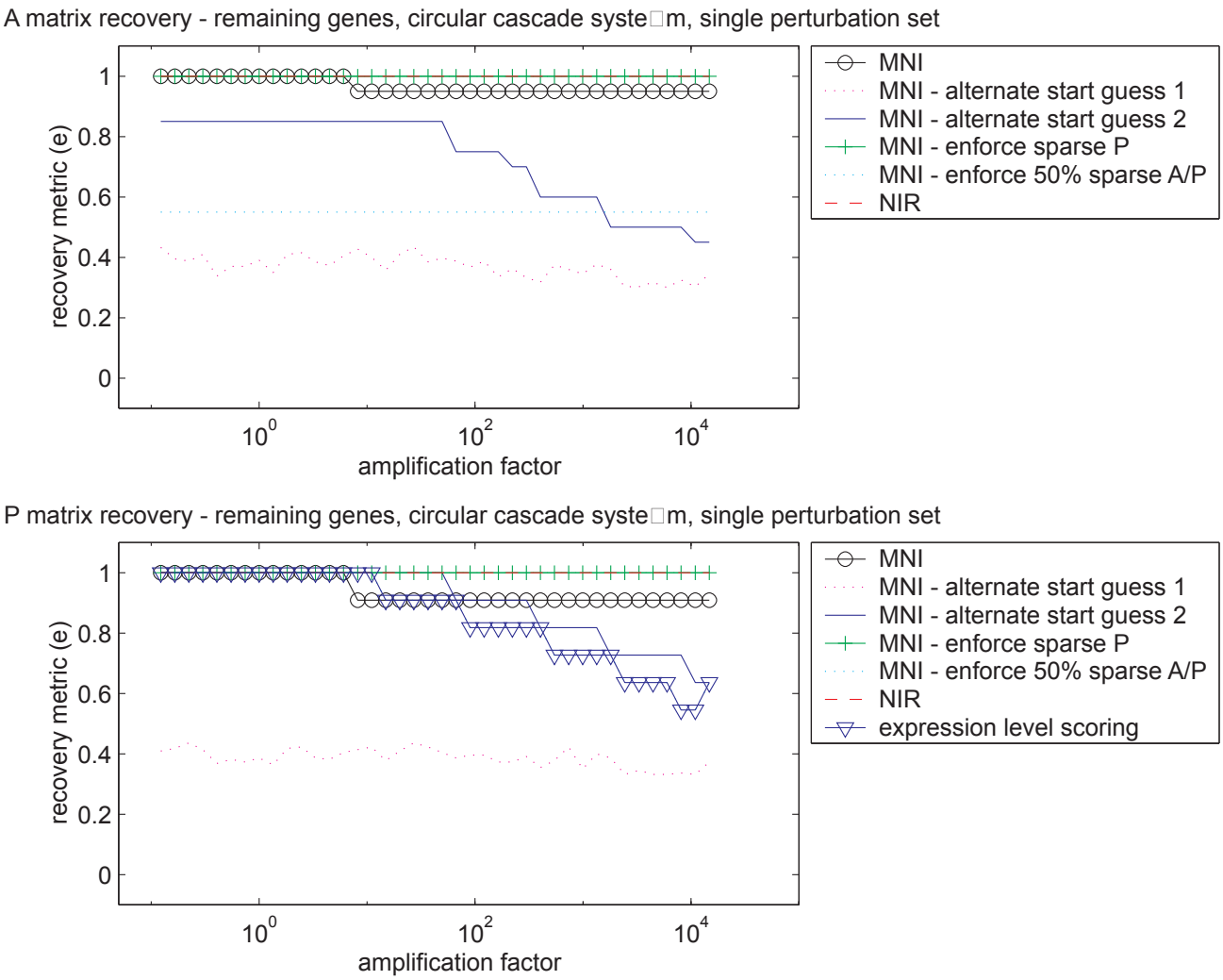

Figure 14: Matrix recovery under increasing amplification factor, single perturbations, circular cascade system. Each graph represents a method or method variant, each point the average recovery metric (e) score over 20 simulated datasets generated using the value of the amplification factor indicated on the x-axis. These results correspond to recovery of every part of the system except gene 1, results for which are instead shown in Figure 15.

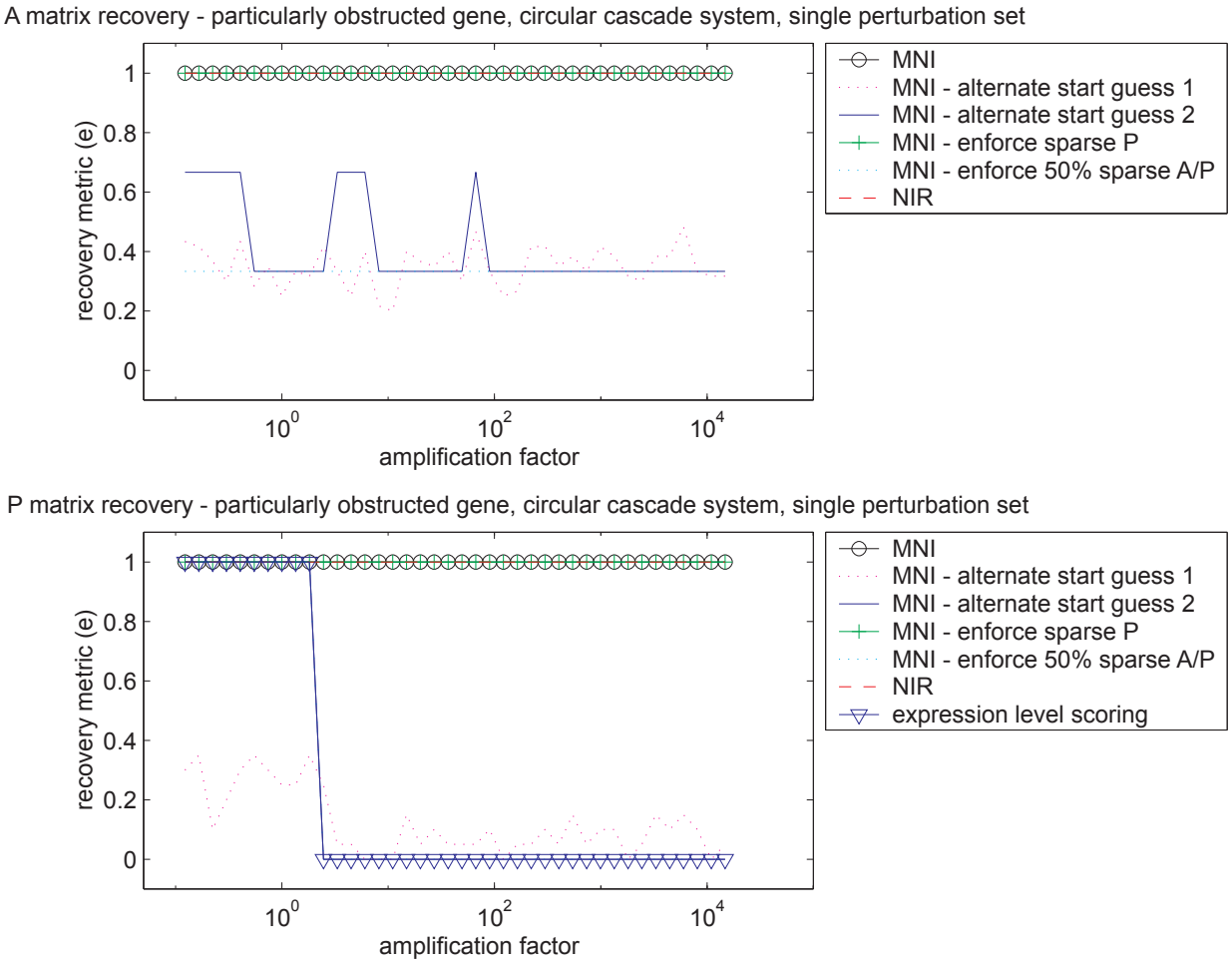

Figure 15: Matrix recovery under increasing amplification factor, single perturbations, circular cascade system. Each graph represents a method or method variant, each point the average recovery metric (e) score over 20 simulated datasets generated using the value of the amplification factor indicated on the x-axis. These results correspond to gene 1 for which we expect results to deteriorate quickly with increasing amplification factor unless the method tested is able to distinguish between direct and indirect responders. For the remaining 11 genes, results are instead shown in Figure 14.

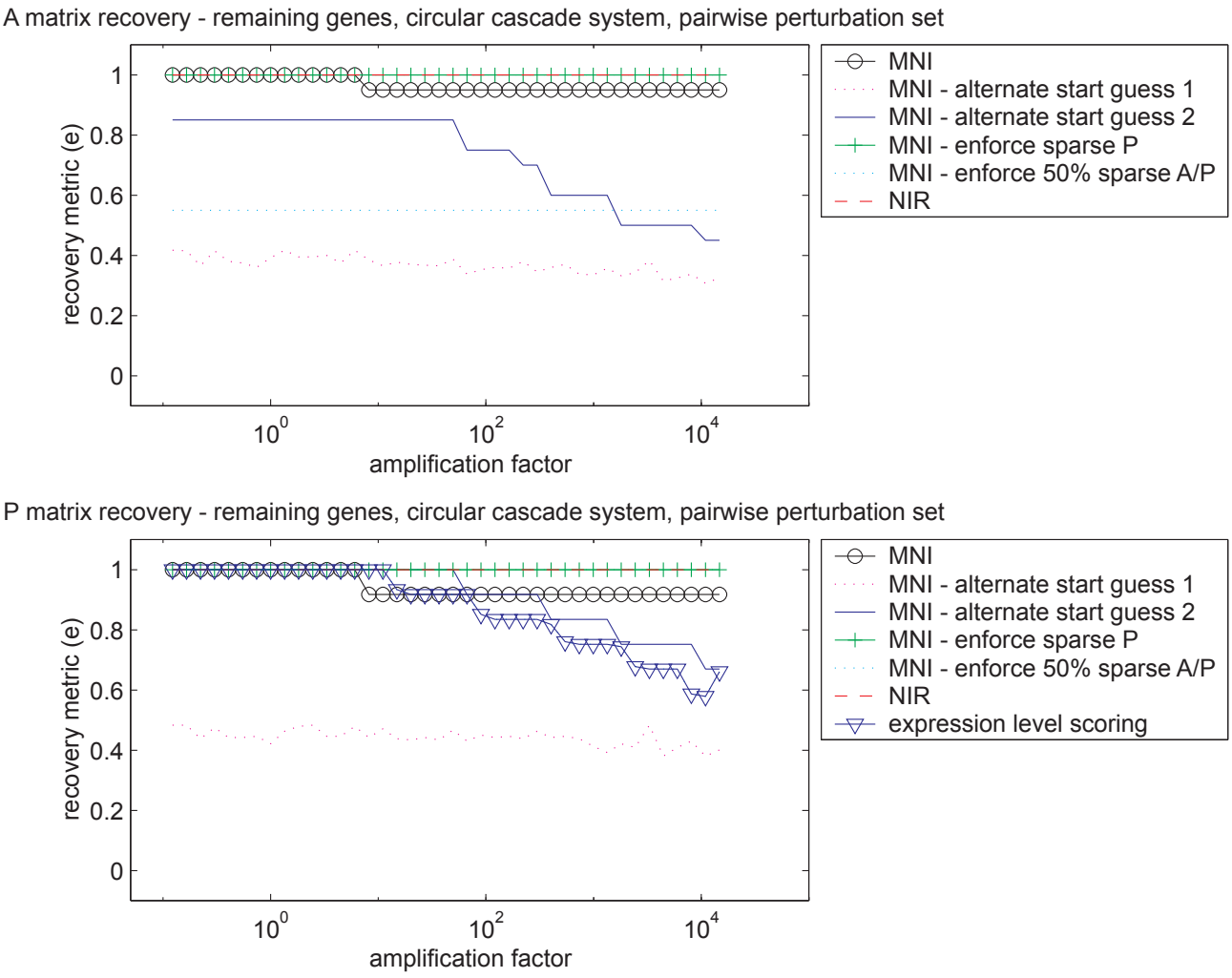

Figure 16: Matrix recovery under increasing amplification factor, pairwise perturbations, circular cascade system. Each graph represents a method or method variant, each point the average recovery metric (e) score over 20 simulated datasets generated using the value of the amplification factor indicated on the x-axis. These results correspond to recovery of every part of the system except gene 1, results for which are instead shown in Figure 17.

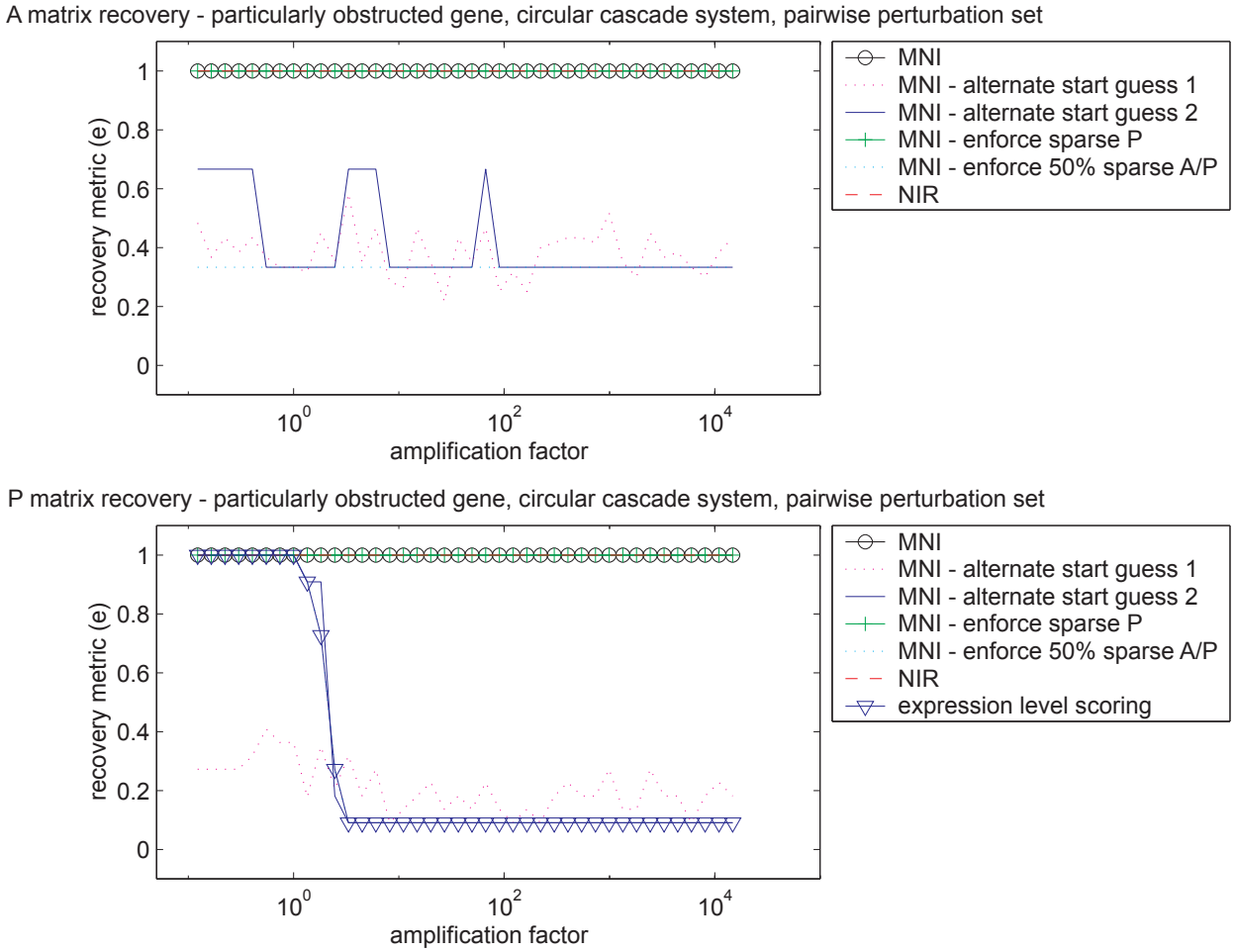

Figure 17: Matrix recovery under increasing amplification factor, pairwise perturbations, circular cascade system. Each graph represents a method or method variant, each point the average recovery metric (e) score over 20 simulated datasets generated using the value of the amplification factor indicated on the x-axis. These results correspond to gene 1 for which we expect results to deteriorate quickly with increasing amplification factor unless the method tested is able to distinguish between direct and indirect responders. For the remaining 11 genes, results are instead shown in Figure 16.

3.4.1 Interpretation

First of all, we can see that, as the amplification factor remains low, system recovery remains possible. There is little difference between the results for the case where we perturb a single gene per experiment and the case where we perturb two genes per experiment; likely because every experiment nevertheless spans the entire gene expression dynamics space. The expected higher performance for MNI/NIR compared to expression level analysis is clearly present for the particularly perturbed gene A. No variant functions significantly better than these methods, nor is NIR significantly better than MNI at recovering the connectivity matrix. There does seem to be an unexpected drop in NIR efficiency at perturbation retrieval at one region of the dataset, possible representing some difficulty in optimizing to the right point for this particular system. Furthermore, we can conclude from the double plasmid-type perturbation set experiment that for a perturbation set that spans the system dynamics sufficiently, it is possible to retrieve perturbations that were not among those used to generate training data for the method.

3.5 MNI σ factor dependence

Figure 18 shows how changing the σ sparsity parameter affects the ability of MNI to recover connectivity and perturbation model parameters from the various systems used here to generate data under otherwise ideal condition.

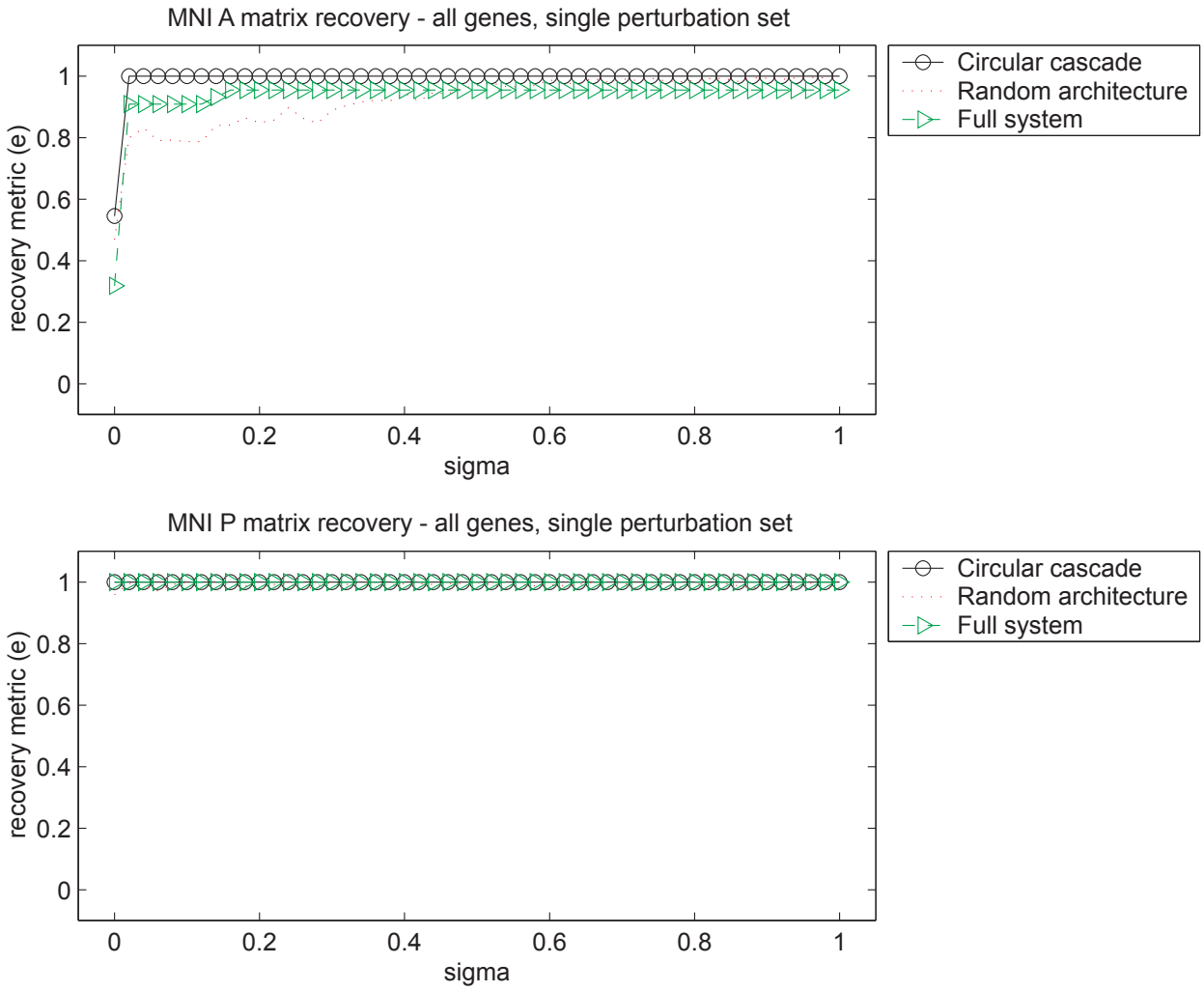

Figure 18: Recovery of connectivity and perturbation matrices using MNI for increasing values of the σ parameter, that is, decreasing assumption of perturbation matrix sparsity. Each graph represents one of the systems presented above, perturbed using the single perturbation set. Each data point represents the average recovery score over 20 simulated datasets.

3.5.1 Interpretation

The choice of $\sigma = 0.25$ made by the creators of MNI obviously work well for some datasets, but less well for others. It is clear that selection of this parameter is relevant, and that increasing it from this value may improve performance in some cases. Setting it too high will tend to make the model less sparse, and as such, more sensitive to over-fitting, and so an approach using some form of cross-validation to determine an optimal σ for a particular dataset may prove useful. A curious observation is that the choice of sigma primarily affects difficulty of connectivity recovery, something that may result from the perturbations being easier than connectivity to recover via these methods.

3.6 Result corroboration MNI - NIR

Applying the MNI algorithm as well as the present implementation of the NIR algorithm to the (biological, controllable plasmid-perturbed) dataset used in the NIR article accomplishes two things. First, any relevant differences between this NIR implementation and the more complex one used in the article will be displayed. Second, performance on this set by MNI can be compared with the results from NIR. The data from the NIR article is given as expression change ratios ($x_i = \frac{y_i - y_{i0}}{y_{i0}}$) rather than absolute concentrations, but still yields a problem on the form $AX = -P$ which is what the implementation of MNI/NIR used in this work requires. Results are displayed as color fields. Figures 19 and 20 show a graphical representation of the matrices as determined in the NIR paper and in this study, while Tables 4, 5, 6, 7 and 8 display the matrices directly.

-0.5970	-0.1790	-0.0100	0.0000	0.0960	0.0000	-0.0110	0.0000	0.0000
0.3870	-1.6700	-0.0140	0.0000	0.0870	-0.0680	0.0000	0.0000	0.0000
0.0440	-0.1890	-1.2750	0.0000	0.0530	0.0000	0.0270	0.0000	0.0000
-0.1808	0.2377	-0.0251	-1.0000	-0.0554	0.0000	0.0000	0.0000	0.3900
0.2810	0.0000	0.0000	0.0000	-2.0940	0.1560	-0.0370	0.0120	0.0000
0.1120	-0.4030	-0.0160	0.0000	0.2050	-1.1470	0.0000	0.0000	0.0000
-0.1710	0.0000	-0.0170	0.0000	0.0250	0.0000	-1.5130	0.0210	0.0000
0.0960	0.0000	0.0010	0.0000	-0.0090	-0.0310	0.0000	-0.4830	0.0000
0.2170	0.0000	0.0000	-1.6780	0.6720	0.0000	0.0770	0.0000	-3.9210

Table 4: Connectivity matrix (A) as determined using NIR in [9].

-1.0000	0.0000	-0.0143	0.0000	0.1503	-0.0381	-0.0234	-0.0007	-0.0575
0.2256	-1.0000	-0.0095	0.0000	0.0389	-0.0428	0.0208	0.0029	0.1174
0.0832	-0.2619	-1.0000	0.0000	0.0601	-0.0053	0.0257	0.0048	-0.0876
0.0709	0.0533	-0.0261	-1.0000	0.0048	0.0646	0.0335	-0.0044	0.3936
1.4102	0.3068	0.0289	0.0000	-1.0000	0.1259	0.0115	0.0051	0.0000
0.1563	-0.4047	-0.0152	0.0000	0.1732	-1.0000	-0.0087	-0.0051	-0.0036
-0.1494	-0.0520	-0.0139	0.0000	-0.0331	0.0398	-1.0000	0.0110	0.3279
0.3041	-0.4157	0.0020	0.0000	-0.0105	-0.0855	0.0134	-1.0000	0.0000
0.1141	-0.2484	0.0049	0.0000	0.1446	0.0092	-0.0106	0.0110	-1.0000

Table 5: Connectivity matrix (A) as determined by MNI from the dataset used in [9].

-0.6915	0.0000	-0.0096	0.0000	0.0991	0.0000	0.0000	0.0000	0.0000
0.5932	-2.2964	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.4293
0.0000	0.0000	-1.2744	0.0000	0.0516	0.0000	0.0000	0.0040	0.0000
0.0000	0.0000	-0.0878	-3.0815	0.0000	0.0000	0.0000	0.0000	1.0541
0.0000	0.0000	0.0550	1.6447	-2.2085	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.2040	-1.1557	0.0000	-0.0085	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	-1.5169	0.0173	0.3387
0.0855	0.0000	0.0000	0.0000	0.0000	-0.0365	0.0000	-0.4832	0.0000
0.0000	0.0000	0.0000	0.0000	0.8903	0.0000	0.0000	0.0568	-5.9419

Table 6: Connectivity matrix (A) as determined by the current implementation of NIR from the dataset used in [9].

0.6529	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	1.1711	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	13.4120	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	1.6705	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	4.5415	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	2.3555	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	4.7083	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	12.8658	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	4.1089

Table 7: Perturbation matrix (P) used to generate the dataset in [9].

0.8927	-0.1347	-0.0000	0.1522	-0.0000	-0.0000	-0.0000	-0.0000	0.0000
0.0000	0.4240	0.0000	0.0138	0.0000	0.0000	-0.0000	-0.0000	0.0000
0.0000	0.0000	10.5097	0.0548	0.0000	-0.0000	0.0000	0.0000	0.0000
0.0000	-0.0000	-0.0000	0.0777	0.0000	0.0000	0.0000	0.0000	0.0000
-1.2333	0.0000	-0.0000	-0.0000	1.6391	-0.0000	0.0000	-0.0000	0.1243
-0.0000	-0.0000	-0.0000	0.1920	-0.0000	1.9672	0.0000	-0.0000	0.0000
0.0000	-0.0000	-0.0000	-0.1002	0.0000	-0.0000	3.0797	0.0000	0.0000
-0.0000	-0.0000	0.0000	-0.0742	-0.0000	0.0000	0.0000	26.6718	0.1261
0.0000	-0.0000	0.0000	0.0270	0.0000	-0.0000	-0.0000	0.0000	0.6813

Table 8: Perturbation matrix (P) from the dataset in [9] as determined by MNI.

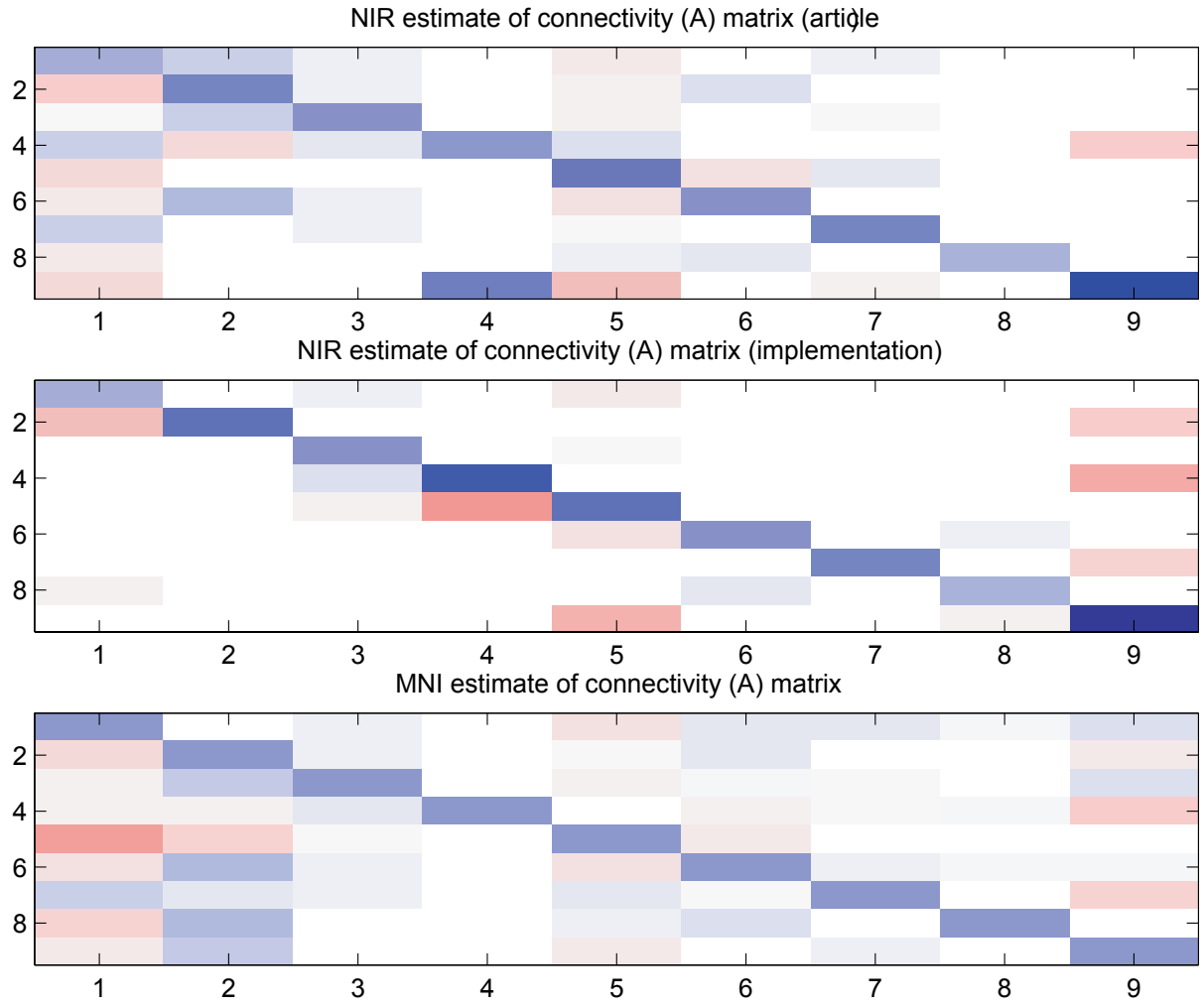

Figure 19: A graphical representation of the connectivity matrices as estimated by NIR in [9], by the implementation of NIR made in this paper, and by MNI. Red fields correspond to positive values, blue to negative. Tables 4, 6 and 5 show these same matrices.

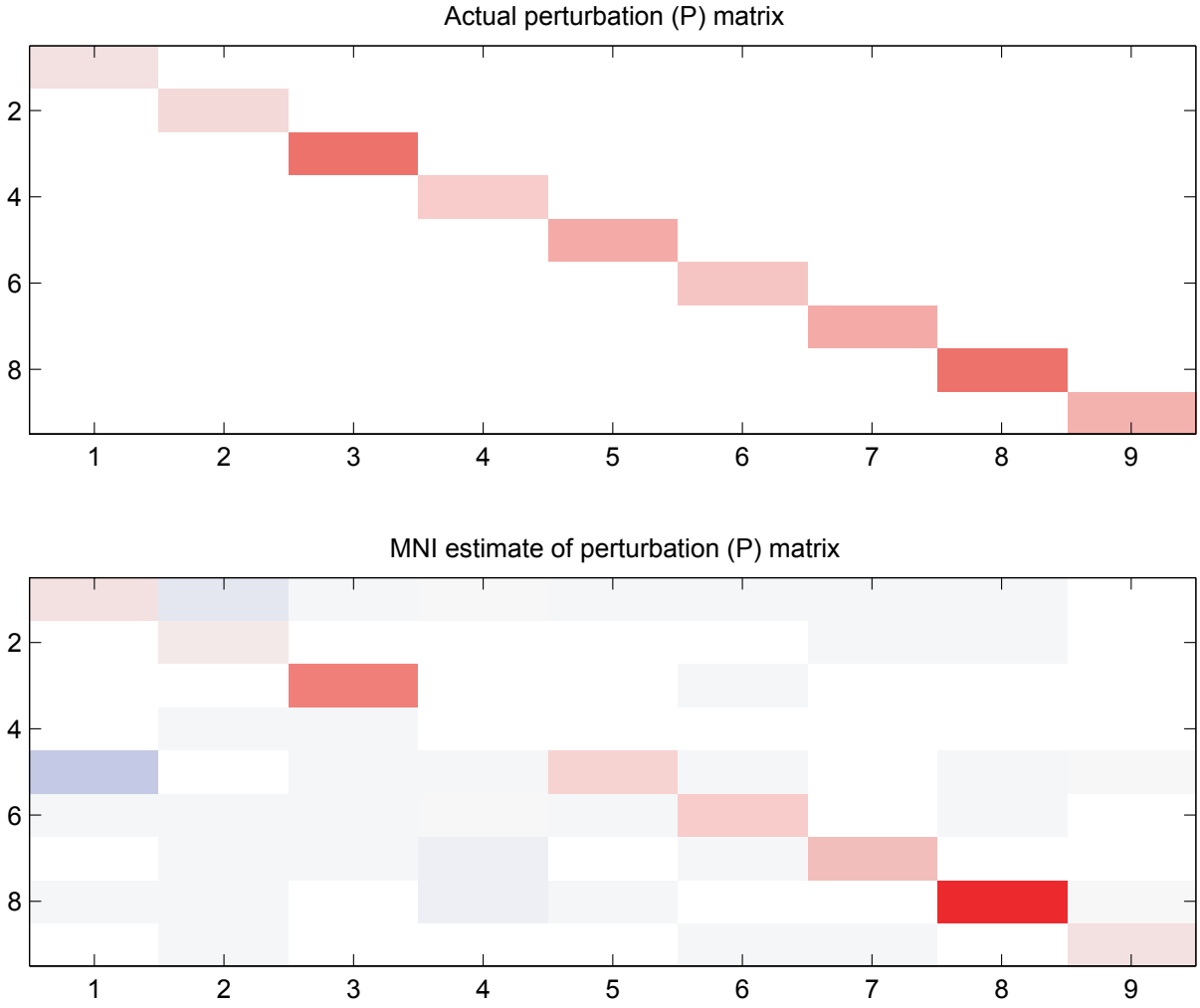

Figure 20: Graphical representation of the true perturbation matrix from [9] (see Table 7) compared to that determined by MNI from the dataset (see Table 8). Red cells represent positive values, blue cells negative.

3.6.1 Interpretation

The model as determined by MNI is similar (although not identical) to that presented in [9], which is significant, particularly as, again, MNI makes use only of expression data whereas NIR also requires perturbation data. The recovered perturbation matrix also captures the main features of the true matrix. Another fact worth noting is that the implementation of NIR used in this work does not reproduce the results of [9] exactly. This would appear to be mainly due to the difference in connectivity selection (cross-validation versus the more complex scheme used in the article). Again, it appears that the cross-validation strategy under these circumstances selects a model ($k = 3$) which is too simple to capture all the traits of this dataset.

4 Discussion

4.1 Overview

This section (4) begins with an overview of its contents (4.1), then discusses the results of the present study, how they may be interpreted and what their limitations are (4.2). The ability of MNI and NIR to distinguish between directly and indirectly responding genes is evaluated (4.2.1) and the various method variants that were proposed are evaluated (4.2.2). Next, reflections are made on the relative performance of MNI and NIR on the same datasets (4.2.3), after which their capacity as a tool for studying gene regulatory networks in themselves is considered (4.2.4). The limitations of this study are adressed (4.2.5), after which some recommendations for further testing and application (4.3) are offered.

4.2 Conclusions

4.2.1 Direct/Indirect responder discrimination

The simulation studies that have been performed aimed primarily at isolating particular characteristics of the techniques evaluated. In particular, the questions we sought to answer were the ability of the MNI method to distinguish between direct and indirect responding genes to a given treatment. From the circular cascade/amplification factor experiments, it is possible to conclude that, for strongly responding indirect targets, both the MNI and NIR methods can succeed where merely considering expression changes fail. However, when considering systems where indirect responders are present (although in moderation) such as the random architecture or the full system, performance was not significantly better than when using expression changes. In the previous studies on biological systems, MNI seemed to outperform expression change analysis, and so a possible conclusion may be that the amplification effect - i.e. system recovery becoming harder as a result of significant indirect responders - has greater influence in actual biology than was assumed in these simulations. A tentative conclusion would thus be that the methods indeed can distinguish between direct and indirect responders.

4.2.2 Variant methods

In none of the cases, our suggested variant methods function significantly better than the previously published methods. There are some indications, however, that varying the value of the MNI σ parameter may affect method performance, and we conclude that this parameter ideally should be selected individually for each dataset.

4.2.3 MNI and NIR comparison

For all the experiments, MNI and NIR functions more or less equally well in recovering system connectivity, and hence also in recovering unknown perturbations. A conclusion will thus be that the introduction of the MNI method removes the need for the NIR method with its cumbersome measurement requirements. While it is difficult to evaluate the performance of either under actual biological conditions, it appears that the extension into MNI is sound and with only small losses in performance.

It is also noted that the number of connections used for the recovered model in NIR is crucial to its performance, and that selection of a suitable number of connections is non-trivial. Cross-validation over simulated datasets like these appear to often select a less complex model than that which is needed to describe the system properly.

4.2.4 Connectivity recovery

Under the conditions used in these experiments, both MNI and NIR are shown to be able to recover system connectivity reasonably well. This is in accordance with earlier simulation studies for NIR and implies that, given datasets that match these conditions (i.e. controllable plasmid experiments, gene deletions or similar directly targetting methods) the methods may be used to study interaction patterns in gene expressions. How well this may work in practice is hard to say.

4.2.5 Limitations

This study evaluates only small systems where perturbation sets are guaranteed to span the gene dynamics sufficiently, effectively matching the experiment setup necessary for NIR with the difference being that we need not know the actual perturbations, only that they possess these properties. However, it appears likely that the results may be generalized to larger systems, particularly if these can be broken down into independent subsystems. The "typical size" of a relevant genetic network is not immediately obvious.

From the MNI article, we can conclude that different types of perturbations are recovered with different ease. The method clearly works best when perturbations have a direct mechanism influencing transcription. How to simulate situations where this is not the case (as with less easily defined perturbations, such as many drugs) is not immediately obvious, but it is clear that the question must be adressed to evaluate the limitations of the methods in such situations.

4.3 Future directions of work

The results from experiments with the Zak network imply that, while the added complexity of going beyond linear dynamics do detract from performance, it does not make it impossible. Whether or not low transcript concentration effects form a significant problem is not obvious at this point, but could be investigated by extending this implementation of the Zak system into including the hybrid stochastic-deterministic approach used in the original article.

Another interesting approach would be to proceed further with a wider range of random architecture simulated systems. System size, degree of sparsity and variance of expression levels would all be interesting factors to increase systematically in much the same way as the experiments performed here, as would be changing the variance of the perturbations in an experiment set or decreasing the rank of the perturbation set. Further exploration of methods to select model complexity is also advised.

Last, a very interesting experiment would be to locate a suitable system with known regulatory interactions in a relatively small network, similar to those simulated here; perturb it using plasmids and compare the MNI connectivity

estimate with the known system. Also, it might be interesting and viable to compile a truly large metaset of published expression experiments for a given system, estimate connectivity from this, then make a small number of controllable plasmid-perturbations and investigate the recovery of these perturbations from this estimate. In this way, the number of experiments necessary to provide more experimental support for the method will decrease drastically.

5 Acknowledgements

I would like to thank my supervisor Mats Gustafsson and the other members of the computational medicine group for giving me this opportunity and for much help along the way, and to Rolf Larsson for accepting the role as scientific examiner of this thesis. Also I would thank the departments of medical genetics and clinical pharmacology for hosting the computational medicine group, and in that providing a welcoming work environment as well as many, many cups of coffee.

References

[1] R. A. Adams. *Calculus: a complete course, fourth edition.* Addison Wesley Longman Ltd, 1999.

[2] T. Akutsu, S. Miyano, and S. Kuhara. Identification of genetic networks from a small number of gene expression patterns under the boolean network model. Pacific Symposium on Biocomputing 4, 1999.

[3] H. Anton and C. Rorres. *Elementary Linear Algebra, applications version.* John Wiley and Sons, 2000.

[4] E. H. Davidson, J. P. Rast, P. Oliveri, A. Ransick, C. Calestani, C.-H. Yuh, T. Minokawa, G. Amore, V. Hinman, C. Arenas-Mena, O. Otim, C. T. Brown, C. B. Livi, P. Y. Lee, R. Revilla, A. G. Rust, Z. J. Pan, M. J. Scilstra, P. J. C. Clarke, M. I. Arnone, L. Rowen, R. A. Cameron, D. R. McClay, L. Hood, and H. Bolouri. A genomic regulatory network for development. *Science*, 295:1669–1678, 2002.

[5] H. de Jong. Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology*, 9(1):67–103, 2002.

[6] D. di Bernardo, T. S. Gardner, and J. J. Collins. Robust identification of large genetic networks. Pacific Symposium on Biocomputing 9, 2004.

[7] D. di Bernardo, M. J. Thompson, T. S. Gardner, S. E. Chobot, E. L. E. andAndrew P. Wojtowich, S. J. Elliott, S. E. Schaus, and J. J. Collins. Chemogenomic profiling on a genome-wide scale using reverse-engineered gene networks. *Nature Biotechnology*, 23:377–383, 2005.

[8] D. di Bernardo, M. J. Thompson, T. S. Gardner, S. E. Chobot, E. L. E. andAndrew P. Wojtowich, S. J. Elliott, S. E. Schaus, and J. J. Collins. Mni source code. Personal communication, 2005.

[9] T. S. Gardner, D. di Bernardo, D. Lorenz, and J. J. Collins. Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*, 301:102–105, 2003.

[10] M. Gustafsson. Dynamical biologial networks 2005: Theory II for exercise 2. Course materials, 2005.

[11] A. J. Hartemink, D. K. Gifford, T. S. Jaakkola, and R. A. Young. Combining location and expression data for principled discovery of genetic regulatory network models. Pacific Symposium on Biocomputing 7, 2002.

[12] R. F. Hashimoto, S. Kim, I. Shmulevich, W. Zhang, M. L. Bittner, and E. R. Dougherty. Growing genetic regulatory networks from seed genes. *Bioinformatics*, 20(8):1241–1247, 2004.

[13] T. R. Hughes, M. J. Marton, A. R. Jones, C. J. Roberts, R. Stoughton, C. D. Armour, H. A. Bennett, E. Coffey, H. Dai, Y. D. He, M. J. Kidd, A. M. King, M. R. Meyer, D. Slade, P. Y. Lum, S. B. Stephanianrs, D. D. Shoemaker, D. Gachotte, K. Chakraburtty, J. Simon, M. Bard, and S. H. Friend. Functional discovery via a compendium of expression profiles. *Cell*, 102:109–126, 2000.

[14] T. E. Ideker, V. Thorsson, and R. M. Karp. Discovery of regulatory interactions through perturbation: Inference and experimental design. Pacific Symposium on Biocomputing 5, 2000.

[15] T. E. Ideker, V. Thorsson, J. A. Ranish, R. Christmas, J. Buhler, J. K. Eng, R. Bumgarner, D. R. Goodlett, R. Aebersold, and L. Hood. Integrated genomic and proteomic analyses of a systematically perturbed metabolic network. *Science*, 292:929–934, 2001.

[16] T. E. Ideker, V. Thorsson, A. F. Siegel, and L. E. Hood. Testing for differentially-expressed genes by maximum-likelihood analysis of microarray data. *Journal of Computational Biology*, 7(6):805–817, 200.

[17] T. I. Lee, N. J. Rinaldi, F. Robert, D. T. Odom, ziv Bar-Joseph, G. K. Gerber, N. M. Hannet, C. T. Harbison, C. M. Thompson, I. Simon, J. Zeitlinger, E. G. Jennings, H. L. Murray, D. B. Gordon, B. Ren, J. J. Wyrick, J.-B. Tage, T. L. Volkert, E. Fraenkel, D. K. Gifford, and R. A. Young. Transcriptional networks in saccharomyces cerevisiae. *Science*, 298:799–804, 2002.

[18] S. Liang, S. Fuhrman, and R. Somogyi. Reveal, a general reverse engineering algorithm for inference of genetic network architecture. Pacific Symposium on Biocomputing 3, 1998.

[19] J. C. Liao, R. Boscolo, Y.-L. Yang, L. M. Tran, C. Sabatti, and V. P. Roychowdhury. Network component analysis: Reconstruction of regulatory signals in biological systems. *Proceedings of the National Academy of Sciences*, 100:15522–15527, 2003.

[20] S. Mnaimneh, A. Davierwala, J. Haynes, J. Moffat, W. Peng, W. Zhang, X. Yang, J. Pootoolal, G. Chua, A. Lopez, M. Trochesset, D. Morse, N. Krogan, S. Hiley, Z. Li, Q. Morris, J. Grigull, N. Mitsakakis, C. Roberts,

J. Greenblatt, C. Boone, C. Kaiser, B. Andrews, and T. Hughes. Exploration of essential gene functions via titratable promoter alleles. *Cell*, 118(1):31–44, 2004.

[21] R. J. Prill, P. A. Iglesias, and A. Levchenko. Dynamic properties of network motifs contribute to biological network organization. *PLoS Biology*, 3(11):1881–1892, 2005.

[22] M. A. Savageau. Design principles for elementary gene circuits: Elements, methods, and examples. *Chaos*, 11(1):142–159, 2001.

[23] I. Shmulevich, E. R. Dougherty, and W. Zhang. Gene perturbation and intervention in probabilistic boolean networks. *Bioinformatics*, 18(10):1319–1331, 2002.

[24] A. Wagner. How to reconstruct a large genetic network from n gene perturbations in fewer than n^2 easy steps. *Bioinformatics*, 17(12):1183–1197, 2001.

[25] L. F. A. Wessels, E. P. van Someren, and M. J. T. Reinders. A comparison of genetic network models. Pacific Symposium on Biocomputing 6, 2001.

[26] K. Wilson and J. Walker, editors. *Principles and Techniques of Practical Biochemistry, 5th edition*. Cambridge University Press, 2000.

[27] M. K. S. Yeung, J. Tegnér, and J. J. Collins. Reverse engineering gene networks using singular value decomposition and robust regression. *Proceedings of the National Academy of Sciences*, 99(9):6163–6168, 2002.

[28] D. E. Zak, G. E. Gonye, J. S. Schwaber, and F. J. D. III. Importance of input perturbations and stochastic gene expression in the reverse engineering of genetic regulatory networks: Insights from and identifiability analysis of an in silico network. *Genome Research*, pages 2391–2405, 2003.