## 7 Supplementary material

**Definition 7.1. Characteristic Function.** *For a scalar random variable $X$, the* characteristic function *is defined as the expected value of $e^{itX}$ where $i$ is the imaginary unit, and $t \in \mathbb{R}$ is the argument of the characteristic function: $\varphi_X(t) = E[e^{itX}] = \int_{-\infty}^{\infty} e^{itx} dF_X(x)$ where $F_X(x)$ is the cumulative distribution function of $X$. If a random variable $X$ has a probability density function $f_X$, then the characteristic function is its Fourier transform, $\varphi_X(t) = \int_{-\infty}^{\infty} e^{itx} f_X(x)dx$.*

### 7.1 Proof of Theorem 3.3

*Proof.*

$$\mathcal{F}(p(x,y)) = \mathcal{F}(\prod_i^*(p(x_i, y_1, \cdots, y_m)) = \prod_i \mathcal{F}(p(x_i, y_1, \cdots, y_m)) =$$

$$= \prod_i \varphi(t_i, s_1, \cdots, s_m) = \varphi(t_1, \cdots, t_n, s_1, \cdots, s_m).$$

□

### 7.2 Proof of Theorem 3.4

*Proof.*

$$\mathcal{F}^{-1}(\varphi(t_1, \cdots, t_n, s_1, \cdots, s_m)) = \mathcal{F}^{-1}(\prod_i \varphi(t_i, s_1, \cdots, s_m))$$

$$= \prod_i^* \mathcal{F}^{-1}(\varphi(t_i, s_1, \cdots, s_m)) \ = \prod_i^* p(x_i, y_1, \cdots, y_m) = p(x,y)\,.$$

□

### 7.3 Proof of Theorem 4.2

*Proof.* The proof follows from the Projection-Slice theorem (also known as the Central Slice theorem) [20, p. 349], which is briefly stated here. Let $f(x,y)$ be a multivariate function and $F(u,v)$ be its matching Fourier transform. Then

$$\mathcal{F}\{f(x)\} = \mathcal{F}\{\int_{-\infty}^{\infty} f(x,y)dy\} = \int_{-\infty}^{\infty} e^{iux}[\int_{-\infty}^{\infty} f(x,y)dy]dx = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{iux} f(x,y)dxdy = F(u,0)\,.$$

This theorem is naturally extended to multiple variables. In our case,

$$\varphi(0,0,t_{i,}\cdots,0) = \prod_j \varphi(t_j, s_1, \cdots, s_m)\Big]_{T\setminus i=0} = \int_{-\infty}^{\infty} e^{it_i x_i}[\prod_j^* p(x_j, y_1, \cdots, y_m)\,]d_X =$$

$$\int_{-\infty}^{\infty} e^{it_i x_i}[\int_{-\infty}^{\infty}\prod_j^* p(x_j, y_1, \cdots, y_m)d_{X\setminus i}\,]dx_i = \mathcal{F}\{\int_{X\setminus i}[\prod_j^* p(x_j, y_1, \cdots, y_m)]d_{X\setminus i}\} = \mathcal{F}\{p(x_i)\}\,.$$

□

### 7.4 Proof of Thm. 4.3

*Proof.* For simplicity, we do not handle the noise variable $z$ in this proof. The noise can be added as a regularization later. We use the linear relation between distributions to extract $X$: $X = A^{-1}Y$. Note that $X$ must distribute according to stable distribution since it is composed from linear combination of stable variables. For the scale parameter we get (using the linearity of $A$ substituted in Prop. 2.1 (a),(b))

$$\gamma_{y_i}^{\alpha} = \sum_j |A_{ij}|^{\alpha}\gamma_{x_j}$$

In vector notation we got

$$\gamma_y^{\alpha} = |A|^{\alpha}\gamma_x\,.$$

Solving this linear system of equations we get

$$\gamma^{\alpha}_{x|y} = (|A|^{\alpha})^{-1}[\gamma^{\alpha}_{y}].$$

Regarding the skew parameter $\beta_x$ using Prop. 2.1(a,b) we get that

$$\beta_{y_i} = \frac{\sum_j \text{sign}(A_{ij})|A_{ij}|^{\alpha}\beta_{x_j|y}\gamma^{\alpha}_{x_j|y}}{\gamma^{\alpha}_{y_i}}\,.$$

In vector notation we get

$$\beta_y = \gamma_y^{-\alpha} \odot [(\text{sign}(A) \odot |A|^{\alpha})(\beta_x \odot \gamma_x^{\alpha})]\,.$$

Now assume that $\gamma_x^{\alpha}$ is a known constant, we can exact $\beta_x$ and get

$$\beta_{x|y} = \gamma_{x|y}^{-\alpha} \odot [((\text{sign}(A) \odot |A|^{\alpha})^{-1}(\beta_y \odot \gamma^{\alpha})]\,.$$

Regarding the location parameter $\delta_x$,

$$\delta_{y_i} = \sum_j A_{ij}\delta_{x_j} + \xi_i\,,$$

$$\xi_i = \begin{cases} \tan(\frac{\pi\alpha}{2})[\beta_{y_i}\gamma_{y_i} - \sum_j \text{sign}(A_{ij})|A_{ij}|\beta_{x_j}\gamma_{x_j}] & \alpha \neq 1 \\ \frac{2}{\pi}[\beta_{y_i}\gamma_{y_i}\log(\gamma_{y_i}) - \sum_j \text{sign}(A_{ij})|A_{ij}|\beta_{x_j}\gamma_{x_j}\log(|A_{ij}|\gamma_{x_j})] & \alpha = 1 \end{cases}.$$

In matrix notation (after some algebra) we get

$$\delta_y = A\delta_x + \xi$$

$$\xi = \begin{cases} \tan(\frac{\pi\alpha}{2})[\beta_y \odot \gamma_y - A(\beta_x \odot \gamma_x)] & \alpha \neq 1 \\ \frac{2}{\pi}[\beta_y \odot \gamma_y \odot \log(\gamma_y) - (A \odot \log(|A|))(\beta_x \odot \gamma_x) - A(\beta_x \odot \gamma_x \odot \log(\gamma_y))] & \alpha = 1 \end{cases}.$$

In total we got a linear system that is solved using

$$\delta_{x|y} = A^{-1}(\delta_y - \xi)\,.$$

□

### 7.5 Proof of Theorem 4.4

*Proof.* W.l.g we prove for the Slice-product algorithm 2(a). The other algorithms are symmetric because the slice/convolution and integral/convolution operations maintain the distributivity property as well.

We are interested in computing the posterior marginal probability

$$p(x_i) = \int_{x\setminus i} p(x,y)d_{X\setminus i} \sim \int_{X\setminus i} p(x_1,\cdots,x_n,y_1,\cdots,y_m)d_{X\setminus i} \tag{7}$$

$$= \mathcal{F}^{-1}\{\prod_i \varphi(t_i,s_1,\cdots,s_m)\Big]_{t_i=0}\}\,. \tag{8}$$

W.l.g assume that $X_i$ is a tree root. Its matching marginal cf $\varphi(t_i)$ can be written as a combination of incoming message computed by the neighboring sub trees:

$$\varphi(t_i) \sim \varphi(t_i,s_1,\cdots,s_m)\prod_{j\in N(i)} m_{ji}(t_i)\,,$$

where the messages $m_{ji}(t_i)$ are defined by the algorithm 2(a). We prove using full induction on the tree diameter. The messages $m_{ji}(t_i)$ satisfy the recursion:

$$m_{ji}(t_i) = \varphi(t_i,s_1,\cdots,s_m)\prod_{k\in N(j)\setminus i} m_{kj}(x_j)\Big]_{x_{j=0}}.$$

The basis for the induction is a tree with a single node $x_1$. In this case there are no incoming messages, $\varphi(t_1) = \varphi(t_i,s_1,\cdots,s_m)$ and we are done. Now assume that the induction assumption

holds for a tree with diameter $d-1$ or less and we want to prove it for a tree with diameter $d$. We make the following construction. We add a new node $x_i$ to the tree to get a tree with diameter $d$. This node has one or more neighbors $j \in N(i)$.

$$\varphi(t_i) \sim \varphi(t_i, s_1, \cdots, s_m) \prod_{j \in N(i)} m_{ji}(t_i) =$$

$$= (t_i, s_1, \cdots, s_m) \prod_{j \in N(i)} p(t_j, s_1, \cdots, s_m) \prod_{l \in N(j) \setminus i} m_{lj}(t_j)] \Big]_{t_j = 0} .$$

Using distributivity of the slice/product (algorithm 2(a)), and the tree assumption (separate trees connected to node $k$ are disjoint), we interchange order of operators to get:

$$\varphi(t_i) \sim \varphi(t_i, s_1, \cdots, s_m)[\prod_{j \neq i} \varphi(t_j, s_1, \cdots, s_m))] \Big]_{t_{j \neq i} = 0} =$$

$$= \prod_i \varphi(t_i, s_1, \cdots, s_m) \Big]_{t_{X \setminus i} = 0}$$

This completes the proof since we have obtained the formulation (8). □

### 7.6 Proof of Theorem 4.5

*Proof.* We start with the scale parameter calculation since it is decoupled from the other parameters.

$$\gamma^{\alpha}_{x_i|y} = \gamma_{y_i} - \sum_{j \neq i} \gamma^{\alpha}_{x_j|y} |A_{ij}|^{\alpha}$$

This iteration is a Jacobi iteration for solving the linear system

$$|A|^{\alpha} \gamma^{\alpha}_x = \gamma_y$$

The linear system solution is given in (4) as desired. It is further known that this iteration converges when $\rho(|R|^{\alpha}) < 1$.

Regarding the skew parameter $\beta$ the Stable-Jacobi update rule is:

$$\beta_{x_i|y} = \beta_{y_i} \gamma^{\alpha}_{y_i} - \sum_{j \neq i} \text{sign}(A_{ij}) |A_{ij}|^{\alpha} \beta_{x_j|y} \, .$$

This conforms to the Jacobi equation for solving the linear system

$$[|A|^{\alpha} \odot \text{sign}(A)] \beta_x = \beta_y \odot \gamma^{\alpha}_y$$

Assuming this system converged, we divide by $\gamma^{\alpha}_x$ to get (4)

$$\beta_{x|y} = \gamma^{-\alpha}_{x|y} \odot [|A|^{\alpha} \odot \text{sign}(A)]^{-1} [\beta_y \odot \gamma^{\alpha}_y] \, .$$

The iteration for computing a skew parameter $\beta$ converges when $\rho(|R|^{\alpha} \odot \text{sign}(R)) < 1$. Using [9, Theorem 8.4.5, Section 8.4] we get that $\rho(|R^{\alpha} \odot \text{sign}(R)|) = \rho(|R^{\alpha}|) > \rho(|R|^{\alpha} \odot \text{sign}(R))$. In other words, when the sufficient condition for the scale parameter $\gamma$ holds ($\rho(|R^{\alpha}|) < 1$), then the skew parameter $\beta$ converges as well.

Now we analyze the shift parameter $\delta$ evolution. The parameter is given by

$$\delta_{x_i|y} = \delta_{y_i} - \sum_{j \neq i} A_{ij} \delta_{x_j|y} - \xi_{x_j|y} \, ,$$

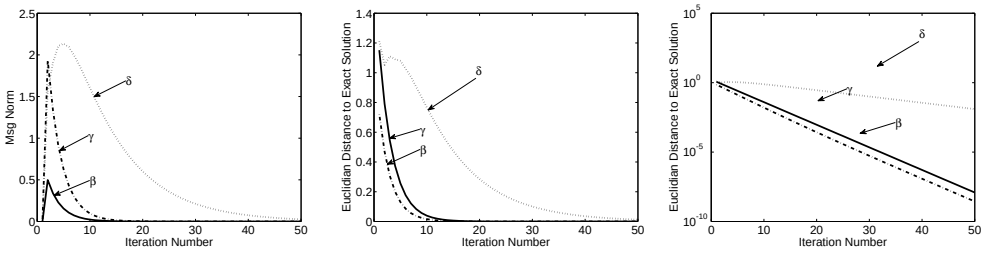
This is a Jacobi equation for solving the linear system

$$A \delta_x = \delta_y - \xi_x$$

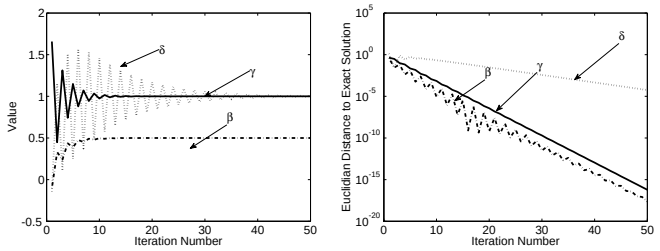
Is given in (4). This iteration converges when $\rho(R) < 1$, which is the second sufficient condition for convergence. □

### 7.7 Synthetic example

We demonstrate the properties Stable-Jacobi, using a small toy example. Experimental settings are borrowed from [32]. The linear transformation matrix is a synchronous CDMA channel transformation with a cross correlation matrix $A_3 = \frac{1}{7}\begin{pmatrix} 7 & -1 & 3 \\ -1 & 7 & 5 \\ 3 & -5 & 7 \end{pmatrix}$. As expected from the convergence analysis, the sufficient conditions for convergence hold since $\rho(|R_3|) = 0.9008 < 1$ and $\rho(|R_3|^{1.5}) = 0.6875 < 1$, and indeed the algorithm converges. We initialized $x = [1, 1, 1]$ and the additive noise $Z_1 \sim \mathcal{S}(1.5, 0, 1, 0), Z_2 \sim \mathcal{S}(1.5, 0.5, 1, 0), Z_3 \sim \mathcal{S}(1.5, 0, 1, 0)$. After computing $p(y)$, we computed $p(x|y)$ using Stable-Jacobi. Regarding convergence dynamics, convergence analysis shows that $\delta$ is converging more slowly since it is dependent on both $\beta$ and $\gamma$. Figure 3(a) shows convergence of message L2 norms. Figure 3(b) plots the Euclidian distance of the intermediate solution on each round (as a vector in $\mathbb{R}^3$) to the exact solution computed by LCM-Stable. This distance indeed goes to zero as expected. Figure 3(c) shows the same distance but using log plot. The almost straight line indicates that the distance is diminishing in a geometric fashion. Unlike the global distance which diminishes monotonically, when examining the second entry of the intermediate solution vector $x_2$ (Figure 3(d)), we see the zigzag behavior which is a well known property of the Jacobi algorithm [2]. This non-monotonic behavior is demonstrated also when examining the Euclidian distance along the single dimension of $x_2$ (Figure 3(e)).




(a) Convergence of message norms (b) Distance to true solution (c) Distance to true solution (log scale)




(d) Posterior marginal of $x_2$ (e) Distance to true solution of $x_2$

Figure 3: Evolution dynamics of the Stable-Jacobi algorithm using CDMA correlation matrix $A_3$. All three parameters $\beta, \gamma, \delta$ are plotted per round, where the parameter of interest is the scale parameter $\delta$ (since i detection is performed by applying the sign operation on it. (a) Convergence of message norm (norm of the current posterior relative to the posterior from previous round) when $\beta \neq 0$. (b) Distance of marginal to true solution (using regular plot). (c) Distance of the vector of marginals for the three users relative to the real transmission (using log plot). (d) Marginal posterior for user 2. The location parameter $\delta$ conforms to the binary transmission. (e) Distance between the posterior of user 2 to the true solution per iteration.

### 7.8 Comparison to previous work

In this section we compare our exact computation of inference in the linear stable model to previous techniques. Since stable distributions do not have a closed-form in the probability domain, any solution deployed in the pdf domain *must involve approximation.* We investigate two existing

approximate inference methods as a reference to our newly developed methods. The first algorithm we implemented is Non-parametric belief propagation (NBP) [29]. NBP works by approximating the observed stable distribution using a Gaussian mixture, and then computes a belief propagation procedure using a linear graphical model [5]. The second algorithm is Expectation Propagation (EP) variant of belief propagation [21], which approximates each Gaussian mixture message using a single Gaussian.

Algorithm 3 lists the approximation methodology we used. Figure 4 depicts the different steps involved. The first tree steps prepare the input to the NBP/EP algorithms by converting the distribution into a mixture of Gaussians, with a relatively low number of mixture components (to allow for efficient execution). We construct a linear model graphical as described in [5]. Then we run NBP/EP for a predefined number of rounds and output the computed belief. Next we can fit stable distribution parameter to the output.

As well known, a drawback of the NBP algorithm is that the number of mixture components grows exponentially when computing the product step of the belief propagation algorithm. To avoid exponential blowup, efficient reduction methods where developed [11, 7]. However, the efficiency comes at the cost of reduced accuracy.

When working even with small problems (tens of variables) both algorithms did not perform well relative to our exact inference method. For example, on a 2D grid graph of 100 hidden nodes and 100 observation nodes, we got an average scale around 0.8146 while the average of true hidden scale was 1. The averages of the skew and shift parameters where even worse, since they are dependent on the scale parameter.

We tried to pinpoint to the root causes of reduced accuracy by constructing a small toy example. We constructed a small graphical model of two hodden nodes $X_1, X_2$ and two observed nodes $Y_1, Y_2$. The hidden node are initialized using a Cauchy distribution with variance 1. To further simplify we set all the edge weights to one, so $A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$. Observations are received using the linear transformation $y = Ax$.

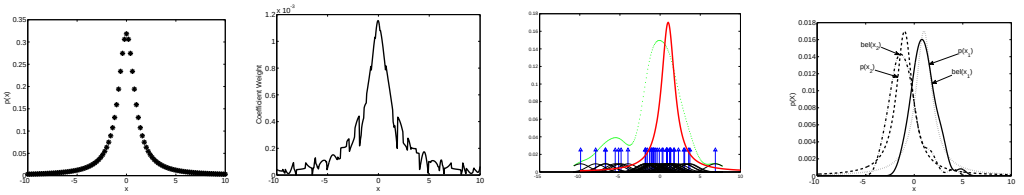
Even for this small problem we can see (Fig. 4(d)) that the NBP output does not match exactly the true solution. We believe that the largest error is rooted in the product step approximation.

Another possible approach is to use Expectation Propagation. EP operates by approximating each Gaussian mixture with a single mixture, creating a light weight and faster approximation (relative to NBP). Fig. 4(e) shows an EP approximation of a single mixture. For Cauchy distributions, EP captured quite well the shape of the distribution (Fig. 4(f)), but less well the exact mean. However, for skewed distributions, EP does not capture well the distribution shape, since the distribution shape can not be approximated using a single Gaussian (Fig. 4(g)).

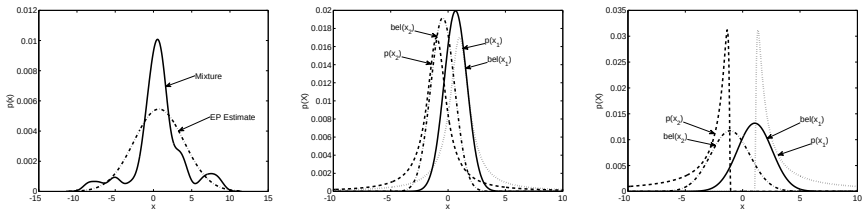
1. Quantisize the stable distribution.
2. Fit a Gaussian mixture to the quantisized observation using Kernel Ridge Regression.
3. Optional: reduce the number of mixture components using sampling techniques.
4. Run non-parametric belief propagation [29] or expectation propagation [21].
5. Quantisize the resulting mixture.
6. Fit a stable distribution to the quantization and retrieve the parameters.

Algorithm 3: Approximate inference for linear stable model.

Overall, we conclude that using previous techniques, it is significantly more difficult to compute inference in a linear-stable model and the results obtained are not accurate. In contrast, using our developed exact inference procedure the solution is obtained exactly by simply computing three matrix inverses.



(a) Step 1: quantization (b) Step 2: fitting a Gaussian mixture (c) Step 3: Resampling (d) Step 4: Output of NBP




(e) Expectation-Propagation estimate (f) Step 4: output of EP (Cauchy prior) (g) Step 4: output of EP (Lévy) prior

Figure 4: Approximated inference using previous techniques: non-parametric BP and Expectation Propagation. In contrast, Stable-Exact computes inference directly in this model by inverting 3 matrices.