# Sequence alignment

See also Durbin *et al.* Chapters 2 and 6

## Objective

- Align the sequences so that the alignment reflects how the sequences evolved

## Why align pairs of sequences?

E.g.

- Comparative genomics

- Modeling evolution (molecular phylogenetics)

- Understanding protein function (conserved amino acid subsequences in a ‘protein family’ suggest functional domains)

## The pairwise alignment problem

Given a scheme for scoring mismatches and gaps find the best-scoring alignment

Can be solved efficiently using Dynamic Programming (see next)

Probabilistic interpretation: Consider two sequences ***x*** and ***y***

Score at a position *i* = $\log\left(\frac{L(x_i, y_i \mid \text{related})}{L(x_i, y_i \mid \text{unrelated})}\right)$

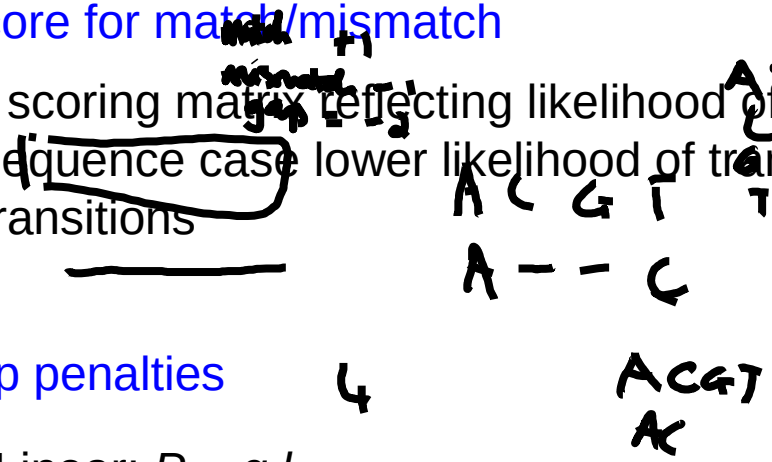
What is the effect of optimizing the alignment score?

Returns the alignment that maximizes the relative likelihood of the related versus unrelated model.

# Alignment scores

## Score for match/mismatch

- scoring matrix reflecting likelihood of mutation types – e.g in DNA sequence case lower likelihood of transversions compared to transitions



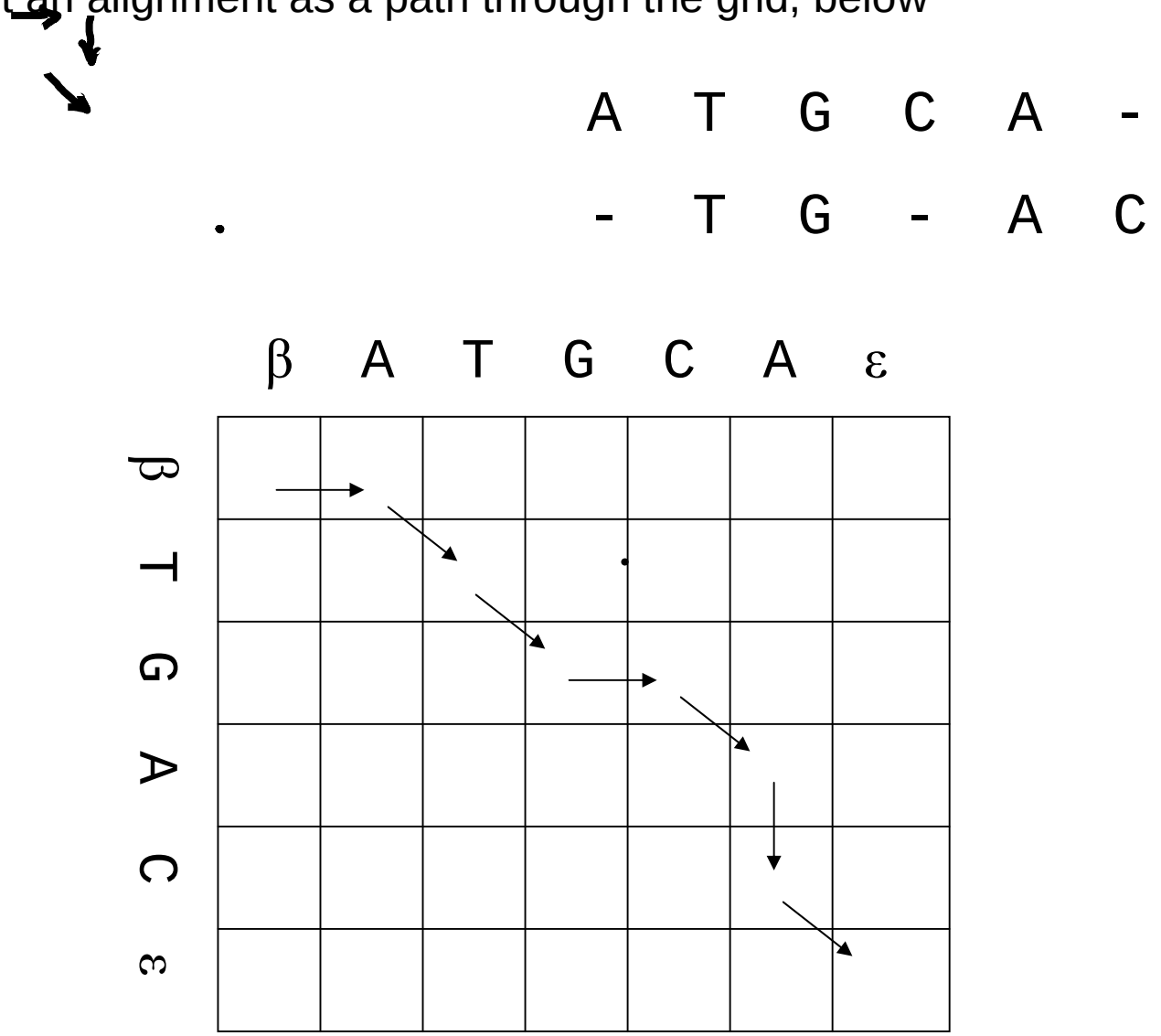
## Gap penalties

Linear: $P = g\,l$

Affine: $P = o + g\,l$

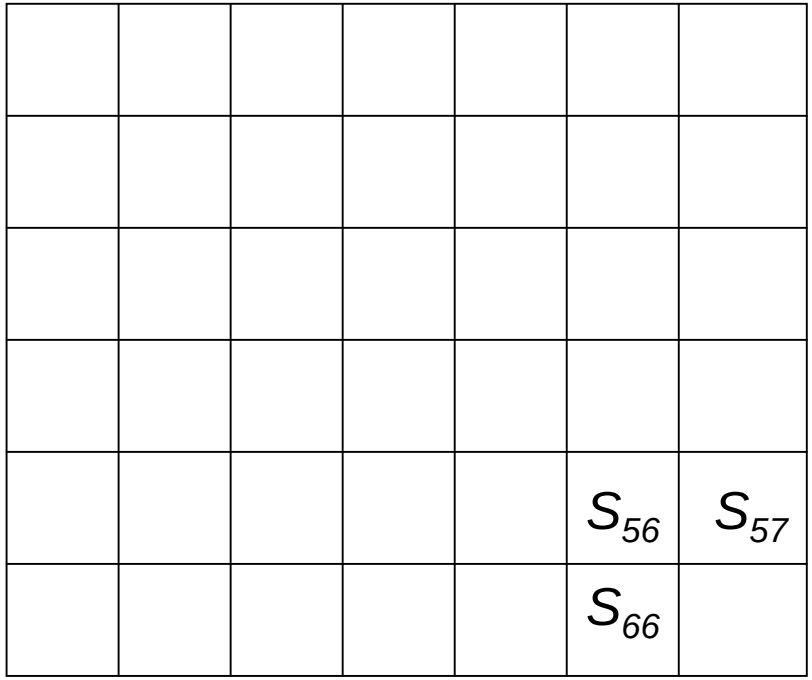
# Dynamic programming algorithm for sequence alignment

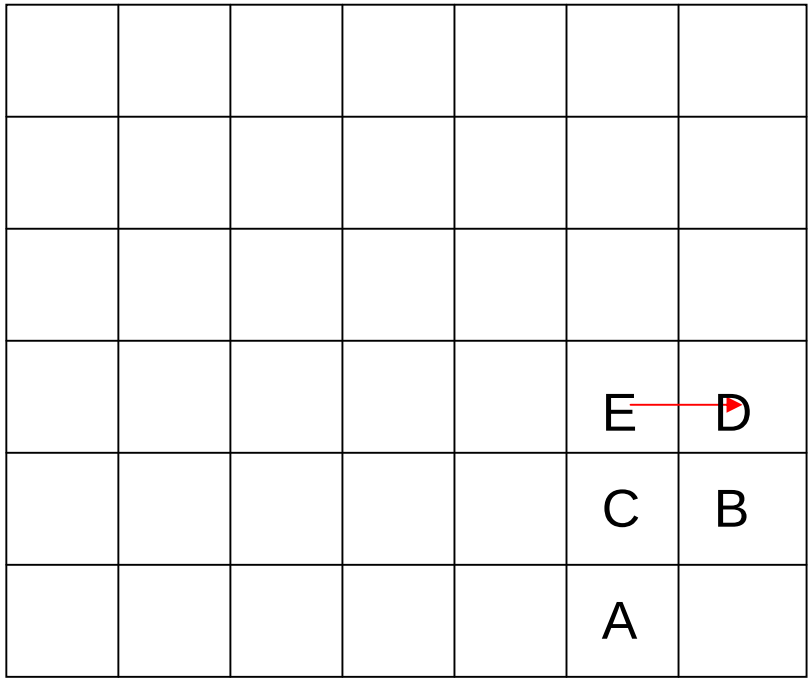
Represent an alignment as a path through the grid, below



Each alignment can be represented by exactly one path through the grid. Finding the optimal path through the grid is the same as identifying the optimal alignment.

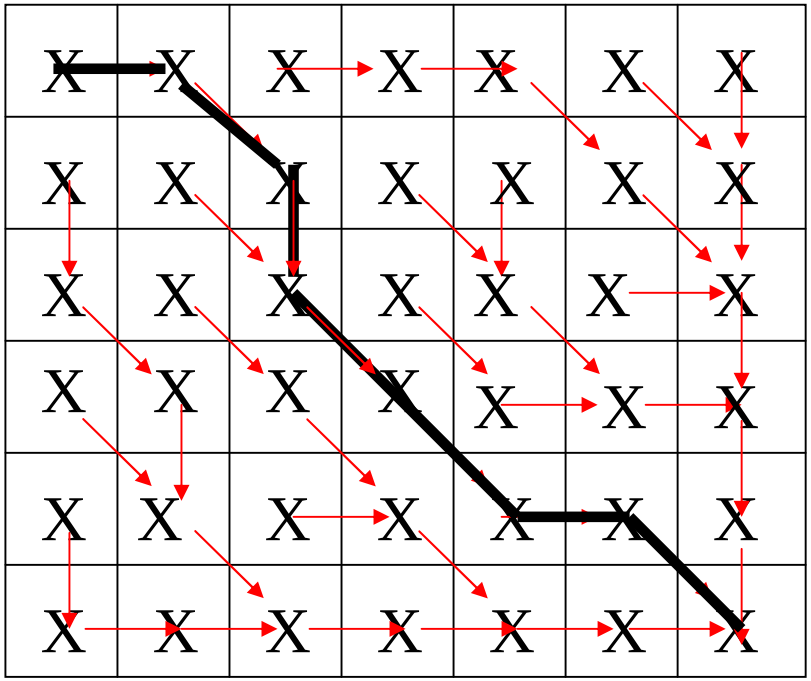
Starting at bottom right, iterate to work out the optimal path from any square *i,j* to the end






We evaluate the three options, write the score of the best scoring path E to the end in cell E and record which was the best choice by placing an arrow as shown

Repeat this process until the grid is complete, then trace the best path from the bottom right to the top left. Convert the path into the corresponding alignment.



Algorithm: Smith-Waterman (optimal sub-path) or Needleman-Wunsch (full path)

Define $s_{i,j}$, the score of the maximal scoring path from position $(i, j)$ to $(L, L)$

*Initialization:* $s_{L-1,L} = s_{L,L-1} = g$

*Induction:*

$$s_{i,j} = \max \begin{cases} 0 \\ s_{i+1,j+1} + M(x_{i+1}, y_{j+1}) \\ s_{i,j+1} + g \\ s_{i+1,j} + g \end{cases}$$

Many software implementations of this algorithm exist

e.g. ClustalX

## Multiple sequence alignment

- Problem is biologically and computationally much more complex

- Typical methods make use of successive pairwise alignment (called progressive multiple alignment)

- Still an active area of research in bioinformatics

## Simply a generalisation of pairwise alignment??

No – because

a. becomes too computationally complex

b. the problem is not quite so straightforward

```
Mouse:  A-TGGA
Rat:    ATT-GA
Human:  ATT-TG
Chimp:  A-TGTG
```

$\not\sim$

```
Mouse:  A-TGGA
Rat:    A-TTGA
Human:  ATT-TG
Chimp:  ATG-TG
```

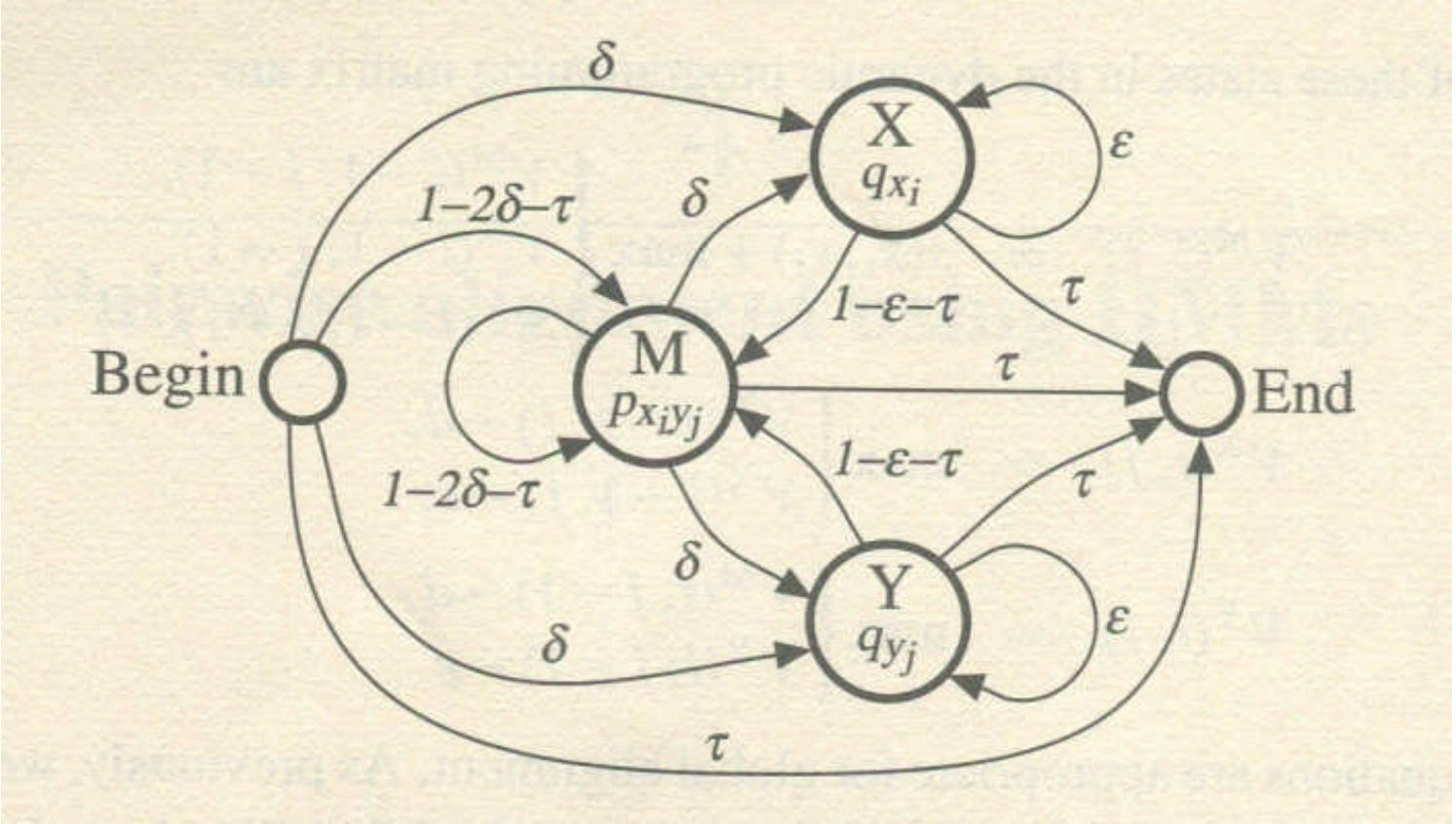
## Aligning multiple sequences: Progressive alignment (Heuristic)

- Perform pairwise alignments
- Align sequences to alignments or alignments to existing alignments
- Do the alignments in some sensible order
  - Most closely related sequences aligned first

## Alignment and phylogeny

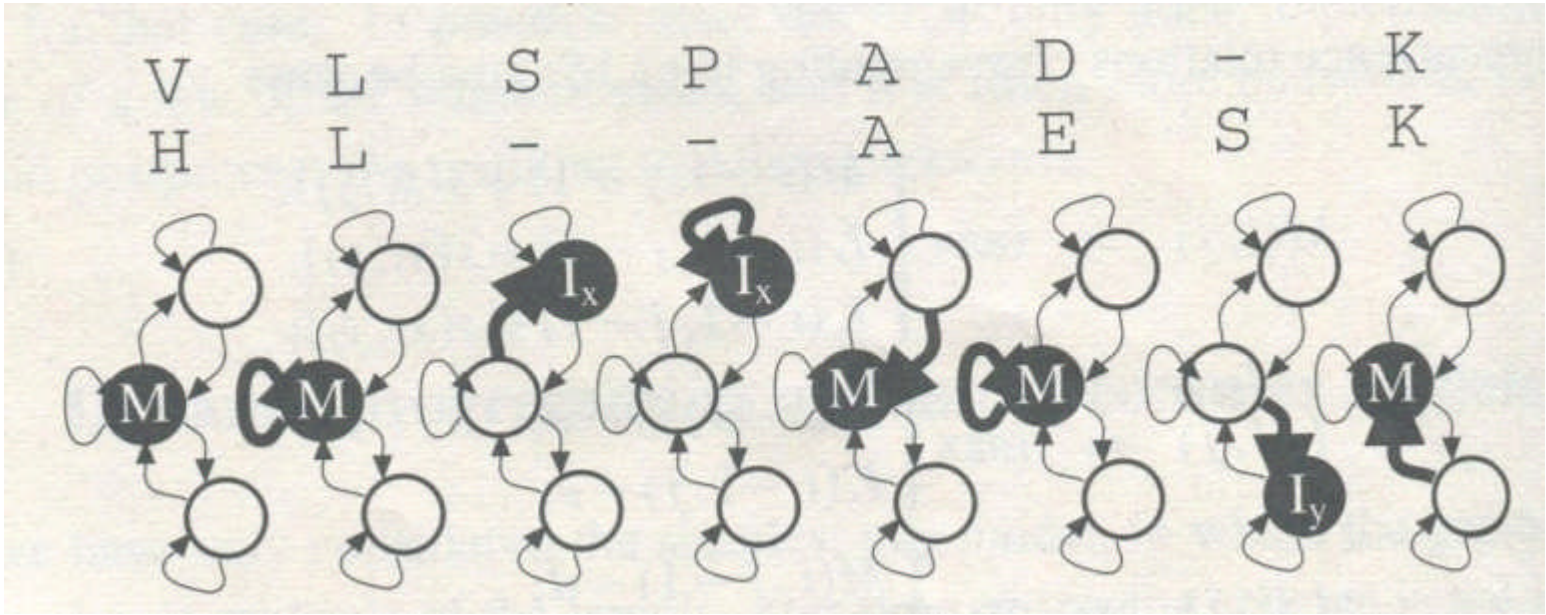
- Phylogenetic tree informs order in which sequences added to alignment
- But tree usually inferred from alignment
- Ideal: joint inference of tree and alignment
- In practice: sometimes tree and alignment inference are iterated
  - i.e. estimate tree; given tree, infer alignment; given alignment, infer tree..

# Probabilistic alignment using Pair HMMs




*Durbin et al. Figure 4.1*

PairHMM generates an aligned pair of sequences



Given a pair of sequences, a path through the hidden states is associated with a defined set of emitted symbol-pairs of the form $(x_{i,}, y_i)$, $(x_i, –)$ or $(–, y_i)$

Therefore, we can work out the joint probability of the path and the sequence pair, or the path that maximizes this joint probability (Viterbi)

Similarly to before, but now with an extra index, define $v_k(i,j)$ as the joint probability of the subpath through the hidden states (ending in state $k$) and the sequence pair up to position $i$ of the first sequence and position $j$ of the second.

## Viterbi algorithm for sequence alignment pair HMM

Define $v_k(i,j) = \max_{\rho_{i,j}} P(x_0 \ldots x_i, y_0 \ldots y_j, \rho_{i,j})$ , where $\rho_{i,j}$ are subalignments ending in state $k$ at position $i$ and $j$ of sequences $x$ and $y$, respectively.

## Viterbi algorithm for sequence alignment pair HMM

Initialisation:

$v_M(0,0) = 1; v_.(0,0) = 0; v_.(0,0) = 0; v_.(-1,j) = v_.(i,-1) = 0$

Recurrence: $i = 0, \ldots, n, j = 0, \ldots, m$

$$
\begin{aligned}
v_m(i,j) &= p_{x_i,y_j} \max \begin{cases} (1-2\delta-\tau)v_M(i-1,j-1), \\ (1-\epsilon-\tau)v_X(i-1,j-1), \\ (1-\epsilon-\tau)v_Y(i-1,j-1); \end{cases} \\
v_X(i,j) &= q_{x_i} \max \begin{cases} \delta v_M(i-1,j), \\ \epsilon v_X(i-1,j); \end{cases} \\
v_Y(i,j) &= q_{y_j} \max \begin{cases} \delta v_M(i,j-1), \\ \epsilon v_Y(i,j-1); \end{cases}
\end{aligned}
$$

We would like to calculate

$$P(x, y) = \sum_{alignments, \rho} P(x, y, \rho)$$

This can be done using a version of the Forward algorithm.

## Forward algorithm for alignment pair HMM

Initialisation: $f_M(0,0) = 1; f_X(0,0) = f_Y(0,0) = 0$

$$f_\cdot(i,-1) = 0; f_\cdot(-1,j) = 0$$

Recurrence: $i = 1,\ldots,n, j = 1,\ldots,m$

$$\begin{aligned}
f_m(i,j) &= p_{x_i,y_j}\,[(1-2\delta-\tau)f_M(i-1,j-1)+ \\
&\quad (1-\epsilon-\tau)(f_X(i-1,j-1)+f_Y(i-1,j-1))] \\
f_X(i,j) &= q_{x_i}[\delta f_M(i-1,j)+\epsilon f_X(i-1,j)] \\
f_Y(i,j) &= q_{y_j}[\delta f_M(i,j-1)+\epsilon f_Y(i,j-1)]
\end{aligned}$$

In this way we can easily calculate

$$P(\rho \mid x, y) = \frac{P(x, y, \rho)}{P(x, y)}$$

This gives the probability that the alignment is correct – usually vanishingly small (but instructive!)

## Backward algorithm for alignment pair HMM

Define $x_i \diamond y_i$ to mean $x_i$ is aligned opposite $y_i$. Then

$$P(x, y, x_i \diamond y_i) = P(x_{1\ldots i}, y_{1\ldots j}, x_i \diamond y_i) P(x_{i+1\ldots n}, y_{j+1\ldots m} | x_i \diamond y_i)$$

Initialisation: $b_M(n, m) = b_X(n, m) = b_Y(n, m) = \tau$
$b_\cdot(i, m+1) = b_\cdot(n+1, j) = 0$

Recursion: $i = n, \ldots, 1, j = m, \ldots, 1 \text{except}(n, m)$

$$\begin{aligned}
b_m(i,j) &= (1 - 2\delta - \tau) p_{x_{i+1}, y_{j+1}} b_M(i+1, j+1) \\
&\quad + \delta [q_{x_{i+1}} b_X(i+1, j) + q_{y_{j+1}} b_Y(i, j+1)] \\
b_X(i,j) &= (1 - \epsilon - \tau) p_{x_{i+1} y_{j+1}} b_M(i+1, j+1) + \epsilon q_{x_{i+1}} b_X(i+1, j) \\
b_Y(i,j) &= (1 - \epsilon - \tau) p_{x_{i+1} y_{j+1}} b_M(i+1, j+1) + \epsilon q_{y_{j+1}} b_Y(i, j+1)
\end{aligned}$$