Require: `pv`, a vector of p-values; `pi1`, either a numeric value for π_1 or the name of the estimation method (default is `"storey"`); `bw`, either a numeric value for the bandwidth h or the name of the estimation method (default is `"sj-dpi"`); `trans`, the name of the transformation function applied to `pv` (default is `"probit"`); `kernel`, name of the kernel function $k()$ used for the density estimation (default is `"Gaussian"`); `pvMin` and `pvMAX`, the truncation limits (default is 0.0 and 1.0, respectively); optionally: `localfdr` a vector of *a priori* knowledge (all `NA` values will be estimated).

```
# PRELIMINARY COMPUTATIONS
trunc = (pv <=pvMin) | (pv>=pvMax) # THE TRUNCATION INDEX
if localfdr is not provided then
  fill localfdr with NA
working=is.na(localfdr) # THE WORKING INDEX
apply the transformation trans to pv to produce x
compute f0 the vector of density f_0 over x
if a numeric value is not provided for pi1 then
  replace pi1 by the estimate of π_1 using the specified method
pi0=1.0-pi1
if a numeric value is not provided for bw then
  replace bw by the estimate of bandwidth h using the specified method
compute q0 and q1 the truncation normalization coefficients q_0 and q_1
build the kernel kords over the 2^k grid (by default, k=8) using the function specified by kernel
# MAIN COMPUTATION PART
draw a random initial vector tau of τ_i
tau[!working]=1-localfdr[!working] # TAKE INTO ACCOUNT a priori INFORMATION
while tau has not converged do
  # PERFORM THROUGH FFT THE APPROXIMATE PRODUCT OF CONVOLUTION
  call the massdist function to distribute x[!trunc] over the grid using the weights tau[!trunc]
  perform the convolution of x by kords using three calls of the fft function
  get the estimation f1 of f_1 over x using a linear interpolation (function approx)
  # UPDATE tau
  index=!trunc & working
  tau[index]= pi1*q1*f1[index]/(pi1*q1*f1[index]+ pi0*q0*f0[index])
# POST-TREATMENT
localfdr[!trunc & working]=1.0-tau[!trunc & working]
plot x vs localfdr along with f0 and f1
```

Output: return `pi1`, `bw`, `f0`, `f1` and `localfdr`

Algorithm 1: Pseudo R code of the kerfdr algorithm.