



```

/*****
Copyright ©2009, Kuberre Systems, Inc. All rights reserved. For more information, address Kuberre Systems, Inc,
805 Turnpike Street, North Andover, MA 01845.
*****/

```

```

package com.kuberre.hpfilter;

import com.kuberre.hansa.jni.HansaJavaAPI;
import com.kuberre.hansa.matlib.HansaMatrix;
import com.kuberre.hansa.matlib.MatrixBuilder;

public class HPF_Hansa {

    public static final short SUCCESS = 0;

    /**
     * Program to apply the "Hodrick-Prescott" filter to time series
     * using a matrix inversion technique. The data is assumed to be
     * organized into a matrix "y" with the rows of y
     * being the observations and the columns being the series.
     * The value of the smoothing parameter is the second argument.
     * HP filtering involves the solution of the difference equation:
     *
     * 
$$y(t) = \text{lam} * g(t+2) - 4 * \text{lam} * g(t+1) + (1+6 * \text{lam}) * g(t) - 4 * \text{lam} * g(t-1) + \text{lam} * g(t-2)$$

     * subject to the two initial conditions:
     * 
$$g(1) - 2 * g(0) + g(-1) = 0;$$

     * 
$$g(2) - 2 * g(1) + g(0) = 0;$$

     *
     * Note that these initial conditions imply that:
     *
     * 
$$y(1) = \text{lam} * g(3) - 2 * \text{lam} * g(2) + (1 + \text{lam}) * g(1)$$

     * 
$$y(2) = \text{lam} * g(4) - 4 * \text{lam} * g(3) + (1 + 5 * \text{lam}) * g(2) - 2 * \text{lam} * g(1)$$

     *
     * and two terminal conditions
     *
     * 
$$y(T-1) = -2 * \text{lam} * g(T) + (1 + 5 * \text{lam}) * g(T-1) - 4 * \text{lam} * g(T-2) + \text{lam} * g(T-1)$$

     * 
$$y(T) = (1 + \text{lam}) * g(T) - 2 * \text{lam} * g(T-1) + \text{lam} * g(T-2)$$

     *
     * If there are a smaller number of observations than series, then
     * there is most likely a mistake: convert the vector/matrix of inputs:
     *
     * @param y - matrix name
     * @param lamda - decay factor
     * @return - value 0 if success, failure otherwise
     */

    public short hpfilter_hansa(HansaMatrix y, double lamda){

```



(-)ansa



```
int r;
int c;
int ny;

// Get number of rows and columns of matrix y
r = y.getRows();
c = y.getCols();

System.out.println("\nrows=" + r + ", cols=" + c);

if(r > c){
    ny = r;
} else {
    ny = c;
}

if(r < c){
    y.trans();// Transpose y
}

System.out.println("\nComputing Hodrick-Prescott Filtered Time Series
with Matrix Inversion using HANSA");
System.out.println("\nGrowth Component is available in HANSA as g");

/* Strategy: Structure difference equation as a matrix equation:
      M g = y
      and then invert M
*/
HansaMatrix M = MatrixBuilder.zeros(ny, ny);

HansaMatrix d1 = MatrixBuilder.ones(ny-2, 1);
d1.mul(lamda);

HansaMatrix d2 = MatrixBuilder.ones(ny-1, 1);
d2.mul(-4*lamda);
d2.setElement(1, -2*lamda);
d2.setElement(ny-1, -2*lamda);

HansaMatrix d3 = MatrixBuilder.ones(ny, 1);
d3.mul(1+6*lamda);
d3.setElement(1, 1+lamda);
d3.setElement(2, 1+5*lamda);
d3.setElement(ny-1, 1+5*lamda);
d3.setElement(ny, 1+lamda);

M = MatrixBuilder.diag( d1, 2);
M.add(MatrixBuilder.diag(d2, 1));
M.add(MatrixBuilder.diag(d3, 0));
M.add(MatrixBuilder.diag(d2, -1));
M.add(MatrixBuilder.diag(d1, -2));

M.inv();// Compute inverse of M
```



(-)ansa



```
M.mul(y);
HansaMatrix g = M;

if (r < c) {
    g.trans();// Transpose g
}

return SUCCESS;
}
}
```