ARTICLE VIII: APPLICATIONS

The **HPS TRANSFORM** takes in a signal and decomposes it into *baseline*, *outlier*, and *error* behavior components. Moreover, let $\check{0}$ represent a time series of (*approximate*) ***zeroes***. Then, generated **HPS approximations** exhibit an (*approximate*) irreducibility property:

$$\begin{array}{lcllll} HPS\left(\langle mon*(i)\rangle \middle| \alpha,m,m'\right) & \approx & \langle mon*(i)\rangle & +\check{0} & +\check{0} & \\ HPS\left(\langle \hat{o}(i)\rangle \middle| \alpha,m,m'\right) & \approx & \check{0} & +\langle \hat{o}(i)\rangle & +\check{0} & \\ HPS\left(\langle \hat{e}(i)\rangle \middle| \alpha,m,m'\right) & \approx & \check{0} & +\check{0} & +\langle \check{f}\left(N\left(0,\sigma^2_{\langle \hat{e}(i)\rangle}\right)\right)\rangle & . \end{array}$$

(8.1)

The **HPS TRANSFORM** is (informally) defined as follows:

$$\boldsymbol{HPS(\langle y(i)\rangle |\ \alpha,\ m,\ m') = \langle mon^*(i)\rangle + \langle o(i)\rangle + \langle e(i)\rangle} \quad \textbf{(8.2)}$$

where ***‹mon*(i)›*** represents a piece-wise constant-level time series approximation to ***‹y(i)›*** (where resultant error is bounded *w.r.t.* confidence level ***α***) and when such fit is *not* feasible, it represents a mean-based tracking of ***‹y(i)›***; ***‹o(i)›*** represents a time-series of heavy-tailed outliers; ***‹e(i)›*** represents a time series of quantization error; and ***m*** and ***m'*** represent CLT-stabilization orders applied to achieve a robust decision-making space.

SECTION 8.1 SOME APPLICATION DOMAINS

Many practical applications exhibit localized stationary conditions (that is, finite bursts of reduced variability and constant mean) and may benefit from the use of stationary-based approximations. There are several possible uses (*not* mutually exclusive) for a stationary-based approximation: **(1)** an *intermediate* representation, **(2)** a *replacement* representation, **(3)** a *co-representation*, and/or **(4)** an *independent* representation for an input signal.

The use of an **HPS approximation** as an *intermediate* representation is suitable for applications that seek to perform a *preliminary operation* (that is, feature generation and extraction) in a domain but with a reduced vocabulary. For example, this makes possible to perform certain kind of (query) operations on this intermediary representation in *lieu* of the original input. The foundation of substring search is the transformation of an input string into a smaller but equivalent signature over which comparison search could be performed more efficiently. This allows translating vast data banks of signals into this intermediary representation over which computationally efficient content searches (*w.r.t.* a reduced vocabulary) could be implemented to obtain *approximate* results. A particular substring search problem of significant interest is the search of DNA sequence matches from large DNA databanks [**REF:MOTIFS**]. An example is developed below.

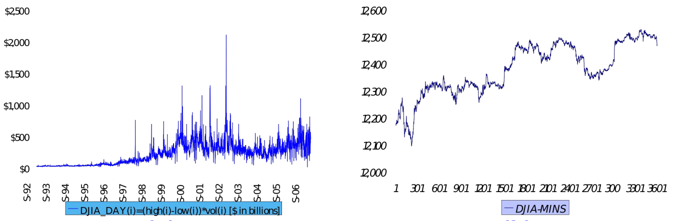
The use of an **HPS approximation** as a *replacement* representation is suitable for applications that seek to *reduce volume of operational data*. For example, based on the reduction property associated with a stationary-based encoding, a motivating example [**NRM:USPTO99**] was introduced in **REQUIREMENTS**, where the **HPS approximation** of a resource monitoring signal was proposed to reduce overhead in both communication and decision-making in order to make possible a class of loosely-coupled distributed "adaptive-process-control" applications.

The use of an **HPS approximation** as a *co-representation* representation is suitable for applications that seek to *augment decision-making power* over an input signal. For example, in our pioneering paper [**NRM:MMCN98**], we introduced preliminary ideas behind the **HPS TRANSFORM** *w.r.t.* the problems of network-bandwidth estimation (and multimedia rate-control) toward the generation of a long-term performance envelope and baseline to drive and complement low-level, short-term estimation (and rate-control) schemes.

The use of an **HPS approximation** as an *independent* representation is suitable for taking advantage of the potential for compressibility of stationary-based approximations. **HPS approximations** are faithful to the original input signal's sampling mean while exhibiting a significant compressibility potential. For example, the **HPS TRANSFORM** decomposition into *baseline*, *outlier*, and *error* components may be of use in analyzing historical performance of random processes.

SECTION 8.2: FINANCIAL TIME SERIES EXAMPLE

Next, we apply our approach to financial time series, a very hard domain where time series are notoriously [**REF:FINANCE**] *non-stationary* and *heavy-tailed*. Nevertheless, we apply the **HPS TRANSFORM** to mine the presence (*if any*) of localized stationary conditions. Specifically, we chose the **Dow Jones Industrials Average** (***DJIA***) [**REF:DJ**]. For comparison purposes, we decided to look at two timescales: *sessions* (DJIA-DAYS) and *intraday* (DJIA-MINS).[1]



(a) **(b)**

Fig. 30: Time plots for the DJIA-DAYS and DJIA-MINS time series.

For DJIA-DAYS, we examined the time interval spanning *September 1992* through *December 2006* – containing approximately ***N=3600*** observations and analyzed the following time series:

$$\boldsymbol{djia\_days(i) = (high(i) - low(i))^* volume(i).} \quad \textbf{(8.3)}$$

This way, DJIA-DAYS estimates profit potential per session. Intuitively, localizes stationary conditions ought to exist within DJIA-DAYS, the amount of money available for investing is *likely* to exhibit finite bursts of such under the ongoing presence of expansion/contraction trends and occasional spikes likely related to world events. **Fig. 30 (a)** shows a time plot of the DJIA-DAYS time series. For DJIA-MINS, we examined data from *December 2006* – containing

[1] The historical session data was obtained from **Yahoo** at *finance.yahoo.com*. The intraday data was obtained from **Wachovia Wealth Management** at *wealth.mworld.com*.

about ***N=3600*** samples. **Fig. 30 (b)** shows a plot of DJIA-MINS. This allows examining the performance of the **HPS TRANSFORM** at two very different timescales, for which the presence of localized stationary conditions have different implications.

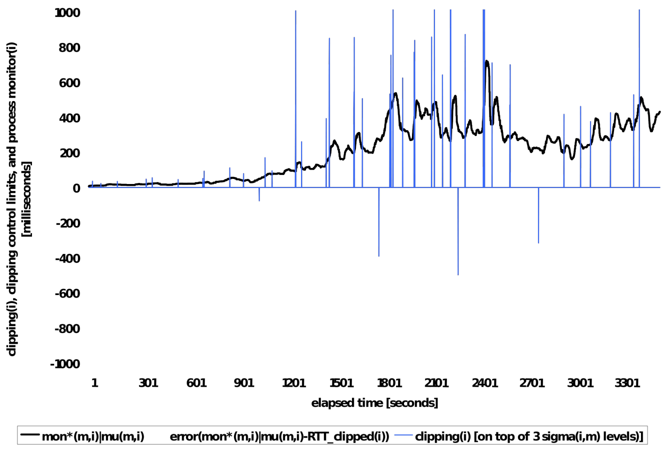


Fig. 31: Resultant HPS approximation at *α=0.005, m=30, and m'=15*.

Next, we apply the **HPS TRANSFORM** using ***α=0.005, m=30, and m'=15*** to each time series. First, for illustration purposes, **Fig. 31** shows the **HPS TRANSFORM**-based decomposition (**8.2**) of the input time series DJIA-DAYS into baseline ***‹mon*(i)›***, error ***‹e(i)›***, and outlier ***‹o(i)›*** components of the resultant **HPS approximation**. Note that outliers are heavy-tailed, **HPS-irreducible** error has *piece-wise* behavior, but so far, the **HPS monitor signal** looks suspiciously like a mean-tracker. To this end, **Fig. 32** provides with a closer look (at the same plot of the **HPS monitor signal** for DJIA-DAYS) but at some random interval – shown within the background of an order *m* CLT stabilization (i.e., sampling mean) of the input signal. The interval ***[300, 1200]*** corresponds to trading sessions between ***11/04/93*** and ***5/29/97***. Now, it is possible to observe that the **HPS TRANSFORM** unearthed bursts of **approximate τ-invariance** and converted them to **ATS segments**, quite heavily within the interval ***(320, 800)*** as well as less frequently elsewhere. However, looking heavily into the interval ***(320, 800)*** one notes that variability among baselines of **ATS segments** within is remarkably small, stable, and centered around the value of **20** (that is, ***$20 billions***). This value represents a *hidden* **HPS fundamental frequency** of the random process being examined (as stated, DJIA-DAYS estimates profit potential per session) and associated with an underlying localized stationary condition that lasted for about ***three months***. Elsewhere, the **HPS monitor signal** defaults to fine-tracking of the aforementioned kind of sampling mean. To this end, **Fig. 33** shows a complete plot of the **HPS monitor signal** along with the computed duration for unearthed **ATS segments** (again, for DJIA-DAYS). Note how the **HPS TRANSFORM** uncovers **ATS segments** of significant duration along localized stationary conditions as well as unearths bursts of **approximate τ-invariance** elsewhere. As stated, localized stationary conditions manifest as concentrations of non-*trivial* **ATS segments**; for example, for DJIA-DAYS, such manifest often and exhibit clusters where segment duration ranges from ***10*** to even ***50*** time units. Moreover, when the DJIA-DAYS time series exhibited significant non-stationary behavior (e.g., from ***1800*** and on), the **HPS TRANSFORM** unearthed short bursts of **approximate τ-invariance** represented by sparse as well as short non-*trivial* **ATS segments**. Finally, insight into the impact of such stationary-based approximation over induced quantization error is given by **Fig. 34.** On this regard, it is worthwhile observing (as preliminary depicted by **Fig. 32**) that despite **HPS quantization**, the **HPS monitor signal** remains *asymptotically* optimal on its tracking of the sampling mean; it is unbiased, precise, and consistent. To this end, **Fig. 34** shows the complete *online* **HPS monitor signal** and corresponding behavior of error (measured in terms of **HPS segment MSE** – that is, error accumulated across the duration of each **ATS segment**). It is worth recalling that the **HPS segment MSE** is autonomously managed (under constant confidence ***α***) for agreement *w.r.t.* the presence of **approximate τ-invariance**. A result of this, as shown, is that the resultant **HPS monitor signal** remains *asymptotically* optimal on its tracking of the sampling mean in spite of induced quantization error during its *approximate* tracking of **HPS fundamental frequencies**.

Now we turn our attention to the DJIA-MINS time series. **Fig. 35** provides with a close look (at the resultant **HPS monitor signal** for DJIA-MINS) at some random interval shown against a sampling mean background as in **Fig. 32**. The chosen interval ***[300, 1200]*** corresponds to intraday quotes during the days of ***12/01/06*** and ***12/12/06***. Again, the **HPS TRANSFORM** unearthed bursts of **approximate τ-invariance** and converted them into **ATS segments**, when feasible, in particular within the interval ***(600, 1000)*** and sporadically elsewhere. Elsewhere, the **HPS monitor signal** defaults to fine-tracking of a (*kind of*) sampling mean. **Fig. 33** shows the complete plot of the **HPS monitor signal** along with the duration of unearthed **ATS segments** for DJIA-MINS. Overall, the **HPS TRANSFORM** unearthed short bursts of **approximate τ-invariance** represented by sparse as well as short non-*trivial* **ATS segments** that ranged in duration to approximately ***30*** time units. Finally, **Fig. 34** shows that the **HPS monitor signal** remained *asymptotically* optimal on its tracking of the sampling mean; it was an unbiased, precise, and consistent.

Now, we take a look at performance metrics that shed insight into the feasibility of generated **HPS approximations**. For the DJIA-DAYS time series, the resultant **HPS approximation** had ***620*** non-*trivial* **ATS segments**, which ranged in duration from ***2*** through ***65*** and were associated with an average **HPS segment duration** of ***4*** time units. As a result, our stationary-based approximation encoding reduced the size of the input signal from ***3600*** *observations* into approximately ***1800*** **HPS forecast** *updates*.[2] However, the **HPS problem** requires that **HPS approximations** be also feasible in terms of error behavior. As stated, targeting accuracy of the **HPS approximation** can be

[2] That is, $\boldsymbol{N\text{-}(\langle n^*\rangle\text{-}1)\cdot\mu_{\langle n^*\rangle}}$, where ***N=3600, ‹n*›=620.***

estimated in terms of the consistency (and behavior[3]) of **z-values** derived from residuals, in this case this being induced **HPS quantization error**. Average ***z-value*** for the $\boldsymbol{\varepsilon_{fast}}$ residual was *only* ***-0.01*** under a standard deviation of *just* ***0.19***.[4] Similarly, tracking accuracy can be estimated in terms of the resultant error (due to a stationary-based model), in this case being resultant **HPS relative error**. Average **HPS relative error** was ***0.66*** under a standard deviation of ***29.83***. Finally, the number of heavy-tailed outliers detected was ***96*** heavy-tail outliers.[5]

For the DJIA-MINS time series,, the resultant **HPS approximation** had ***140*** non-*trivial* **ATS segments**, which ranged in duration from ***2*** through ***31*** and were associated with an average **HPS segment duration** of ***6*** time units. As a result, our stationary-based approximation encoding reduced the size of the input signal from ***3600*** *observations* into approximately ***2800* HPS forecast** *updates*.[6] Average ***z-value*** for the $\boldsymbol{\varepsilon_{fast}}$ residual was *only* ***-0.01*** under a standard deviation of *just* ***0.19***. Average **HPS relative error** was ***0.66*** under a standard deviation of ***29.83***. Finally, the number of heavy-tailed outliers detected was ***96*** heavy-tail outliers.

In summary, the resultant **HPS approximations** (under *default* parameters) unearthed significant amount of **approximate τ-invariance** from the DJIA-DAYS and DJIA-MINS time series. The results and behavior of **HPS approximations** were consistent with findings from our baseline experiment. Moreover, we introduced a new kind of robust feature-extraction – applicable for forensic analysis, investment analysis, etc. – on one of the hardest time series domains.[7] The presence (*or absence*) of **approximate τ-invariance** represents *very* valuable decision-making knowledge on this application-domain, in particular when such findings are provided under a robust decision-making model exhibiting known error bounds and confidence – as in our case.

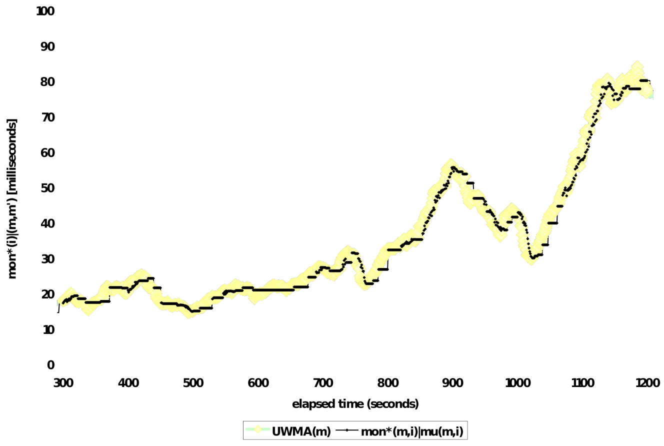


Fig. 32: Close look at a random interval for the resultant HPS monitor signal (for DJIA-DAYS) shown within an (order *m*) sampling mean for the input signal. The interval [300, 1200] corresponds to trading sessions between 11/04/93 and 5/29/97.

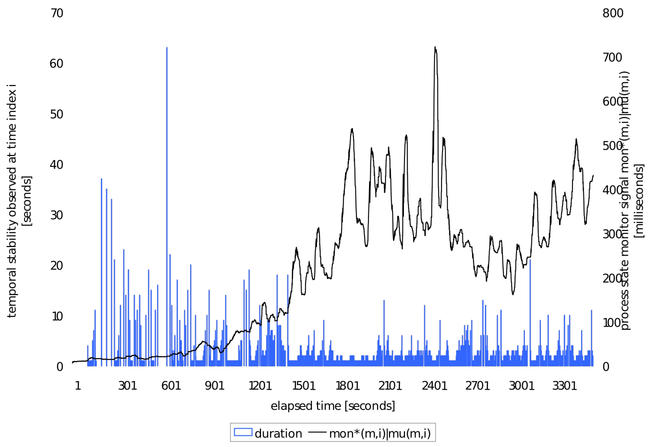


Fig. 33: Resultant HPS monitor signal (for DJIA-DAYS) together with corresponding HPS segment duration for each non-*trivial* (and *trivial*) ATS segments.

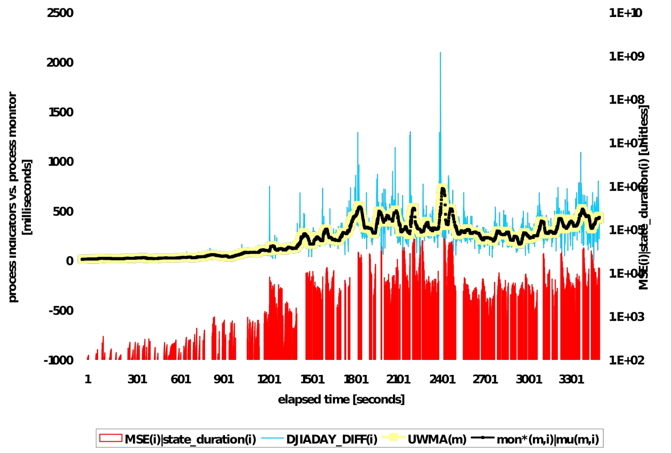


Fig. 34: Resultant HPS monitor signal (shown within the background of an unconstrained sampling mean estimator) (for DJIA-DAYS) together with corresponding HPS segment MSE for each non-*trivial* (and *trivial*)ATS segments, and resultant HPS relative error.

[3] Although not shown, z-values were *approximately* normally distributed under ***N(0.01, 0.19²)*** as expected.

[4] Recall that measurements are in ***billion*** dollars, where the range of values from the input signal was approximately from ***1*** to ***65*** billion dollars (see **Fig. 31**).

[5] Outlier detection was based on ***six-sigma*** limits based on order ***30*** CLT-stabilization delay. Note that this delay is customizable in a manner that is independent from the parameters of **HPS decision-making**.

[6] That is, $\boldsymbol{N-(\langle n^*\rangle-1)\cdot\mu_{\langle n^*\rangle}}$, where $\boldsymbol{N=3600,\ \langle n^*\rangle=620}$.

[7] These time series are known to be non-stationary, heavy-tailed, and composed of an infinite number of random sources laid across multiple as well as hidden timescales.

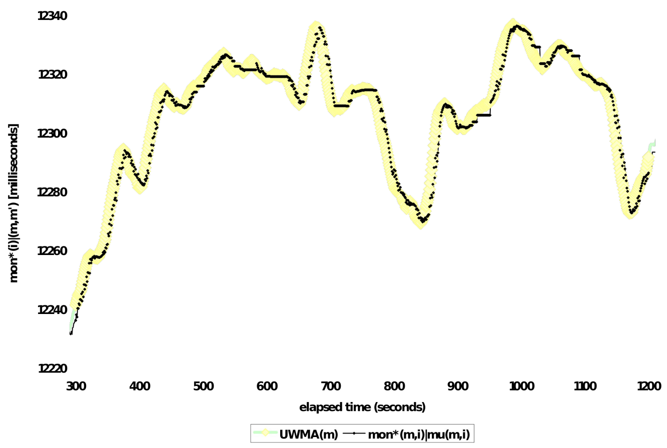


Fig. 35: Close look at a random interval for the resultant HPS monitor signal (for DJIA-MINS) shown within an (order *m*) sampling mean for the input signal. The interval [300, 1200] corresponds to intraday data between 12/01/06 and 12/12/06.

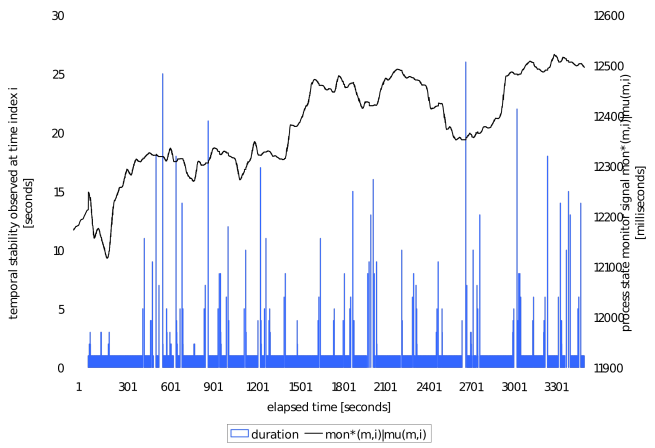


Fig. 36: Resultant HPS monitor signal (for DJIA-MINS) together with corresponding HPS segment duration for each non-*trivial* (and *trivial*) ATS segments.

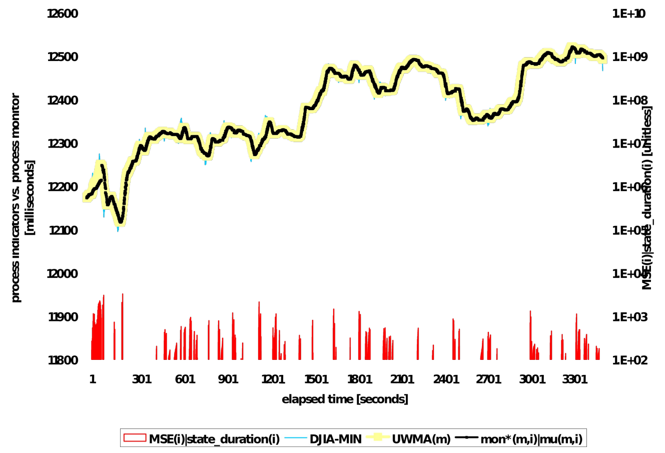


Fig. 37: Resultant HPS monitor signal (shown within the background of an unconstrained sampling mean estimator) (for DJIA-MINS) together with corresponding HPS segment MSE for each non-*trivial* (and *trivial*)ATS segments, and resultant HPS relative error.

SECTION 8.3: A STENOGRAPHY EXAMPLE

The following stenography[8] example application sheds ***SNR*** insight into the **HPS TRANSFORM** *wrt* mean tracking. Our goal is to hide a message encoded as a sequence of process states within an apparent noise process. The encoding is achieved as follows; the mean of each such process state represents a single letter. Through the use of the **HPS TRANSFORM** we can increase the ability to data-mine such process states (with respect to the presence of inserted or not noise).

The coding scheme used was simply a substitution cipher ((B G E A O F S H J R ...) → (-13 -12 -11 -10 -9 -8 -7 -6 -5 -4 ...)) [**SUBSCIPHER**], for which the letters with the highest frequencies were assigned (plus/minus) integers closer to 0 whereas the rest have assigned values in the outer range of the coding interval. The initial alphabet was coded into an integer range of -13 and 13. This coding was disguised through the use of a noise dispersion spanning the interval -115 and 115 (i.e., an order of magnitude toward each direction). The plaintext was just the string: "*steno doyou thinkit shouldbe illegal tosend textmessages anddrive end*".

We chose a particularly well-behaved form of noise ***PHI(*i*)*** where every other sample was sampled from a uniform-random space but its successor sample cancelled out its contribution as follows:

```
PHI(i)= if (*i* is even) contribution(*i*)=random()
        else contribution(*i*)=-contribution(*i*-1).
```

The resulting input signal presented to the **HPS TRANSFORM** is shown in **Fig. X**. Note that the input signal seems to be apparent noise – even when underneath such there is a text message present.

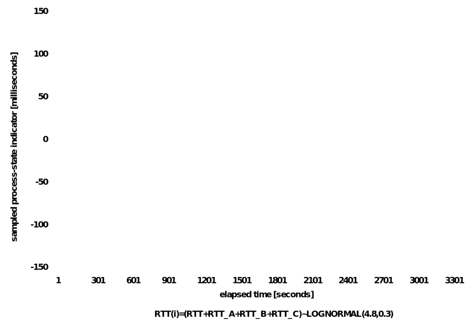


Fig. X: Input signal to the stenography example.

Shared knowledge ***S=(y(i), L→N, S_{min})*** between transmitter and receiver consisted of: (1) the **input signal *y(i)***, (2) the **letter-to-integer** coding scheme ***L→N***, and (3) a **critical threshold *S_{min}*** representing the expected minimum duration of HPS states.

Encoding and decode were done in LISP. The resulting time series had 10000 samples[9] which were presented to the simulator, which subsumed the series under an additive noise process. To extract this message, a

[8] It is emphasized that this is a stenography (not cryptography) example; in actual fact, it is **not** assumed that the coding scheme has not been compromised.

[9] Under a sampling arrangement of 10^4 samples/s, 1s is needed to convey the original 50 input characters.

robust application of the **HPS TRANSFORM** was applied with parameters ***HPS(60,30,0.001)*** corresponding to a shared knowledge about the minimum duration of said states. Fig. X shows the results of the application of the **HPS TRANSFORM** to the input signal. The **HPS TRANSFORM** unearths the presence of a series of process states, extracted from this apparent noise process.

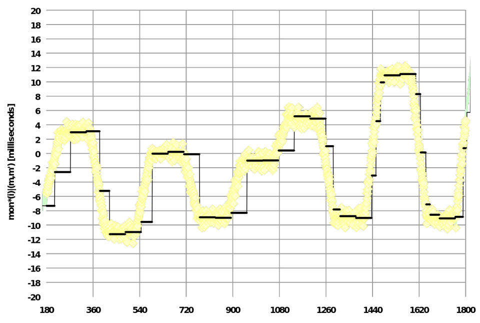


Fig. X: Resulting HPS transform of the input signal shown within the context of mean tracking.

After applying the transform, the first 3600 samples of the simulator's output were then presented to a decoder implemented in LISP. **Table X** shows the decoded output.

S-START	S-END	S-DURATION	S-MEAN	LETTER
97	212	115	-7	S
273	386	113	3	T
424	546	122	-11	E
589	772	183	0	N
772	894	122	-9	O
955	1077	122	-1	
1138	1260	122	5	D̄
1315	1437	122	-9	O
1486	1608	122	11	Y
1663	1791	128	-9	O
1829	1977	148	13	U
2014	2197	183	-1	
2197	2319	122	3	T̄
2380	2502	122	-6	H
2563	2685	122	-3	I
2746	2868	122	0	N
2929	3051	122	7	K
3112	3234	122	-3	I
3295	3417	122	3	T
3417	3600	183	-1	_

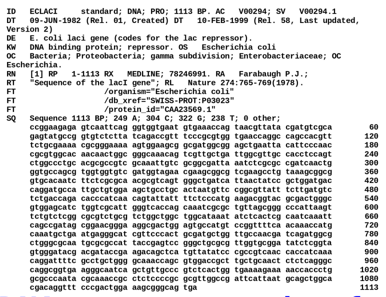
Fig. X: HPS OUTPUT FROM LISP

Effectively, the signal was successfully encoded into variable length states, for which their respective means represented the coded value of letter of the plaintext, hidden through one order of magnitude additive noise with built-in disguised noise cancellation, the plaintext was remained completely recoverable. Moreover, under such large magnitude of noise, successful decoding of the message can only be achieved on the possession not only of the awareness of the **HPS TRANSFORM** but also the **critical threshold** S_{min}. For completeness, it is stated that this cryptographic component of this code (i.e., the **letter-to-integer** coding scheme ***L→N***) could be compromised if the message represents a writing works that is *unusually/sufficiently* long to derive the relative frequency of (somehow) quantized values present in the input signal.

SECTION 8.3: DNA EXAMPLE

Input signals analyzed so far took values from ℜ, for which the HPS TRANSFORM generated an **HPS approximation** whose values were also in ℜ. We apply next the HPS TRANSFORM to a ***DNA*** time series – a *categorical* series, which takes values from a small set ***Ω={A,C,G,T}***).

This example illustrates data conditioning for categorical data as there are many ways to represent categorical tokens, yet only few are ***well-behaved*** *w.r.t.* statistical testing purposes. First, ***DNA*** data is *categorically* coded – and in particular – over a very *small* value set ***Ω*** (where ***Ω={A,C,G,T}***); therefore, a coding scheme ***Ω→ℜ*** is needed. Clearly, to prevent introducing unknown bias into **HPS decision-making**, the coding scheme must provide a mapping (from ***Ω*** to ℜ) exhibiting uniform and unbiased distribution over some mapping interval. Our second concern is less obvious; the *size of the original alphabet* ***Ω*** (that is, ***4***; corresponding to the four ***DNA*** tokens) is *too* small. For robustness in **HPS decision-making**, it is desirable that the coding scheme ***Ω→ℜ*** generates a *sufficiently large set of values* in ℜ (that is, more that just ***four*** as long as such does not introduce unwanted bias across generated values). Third, due to the nature of a stationary-based approximation, if the *stepsize* of the coding scheme ***Ω→ℜ*** is *too* small, it is likely that **approximate τ-invariance** will be *generated* (as opposed to *unearthed*) from ℜ (as encoded differences loose their relative strength); therefore, we want that a *sufficiently large step-size* separates generated values in ℜ. Finally, an implementation concern also needs to be addressed. If repeated values are present during the start of the time series, crucial initialization of variability indicators may be affected. To avoid this, a *very small* noise function is introduced to values generated by the coding scheme. As long as such conditioning is small enough in relative size to the *stepsize* of the coding scheme, by virtue of its (*statistical*) nature, **HPS decision-making** would remain impervious to such negligible variability component. One may note that even non-categorical time series may benefit from applying the aforementioned conditioning.



```
ID   ECLACI     standard; DNA; PRO; 1113 BP. AC   V00294; SV   V00294.1
DT   09-JUN-1982 (Rel. 01, Created) DT   10-FEB-1999 (Rel. 58, Last updated,
Version 2)
DE   E. coli laci gene (codes for the lac repressor).
KW   DNA binding protein; repressor. OS   Escherichia coli
OC   Bacteria; Proteobacteria; gamma subdivision; Enterobacteriaceae; OC
Escherichia.
RN   [1] RP   1-1113 RX   MEDLINE; 78246991. RA   Farabaugh P.J.;
RT   "Sequence of the lacI gene"; RL   Nature 274:765-769(1978).
FT                   /organism="Escherichia coli"
FT                   /db_xref="SWISS-PROT:P03023"
FT                   /protein_id="CAA23569.1"
SQ   Sequence 1113 BP; 249 A; 304 C; 322 G; 238 T; 0 other;
     ccggaagaga gtcaattcag ggtggtgaat gtgaaaccag taacgttata cgatgtcgca        60
     gagtatgccg gtgtctctta tcagaccgtt tcccgcgtgg tgaaccaggc cagccacgtt       120
     tctgcgaaaa cgcgggaaaa agtggaagcg gcgatggcgg agctgaatta cattcccaac       180
     cgcgtggcac aacaactggc gggcaaacag tcgttgctga ttggcgttgc cacctccagt       240
     ctggccctgc acgcgccgtc gcaaattgtc gcggcgatta aatctcgcgc cgatcaactg       300
     ggtgccagcg tggtggtgtc gatggtagaa cgaagcggcg tcgaagcctg taaagcggcg       360
     gtgcacaatc ttctcgcgca acgcgtcagt gggctgatca ttaactatcc gctggatgac       420
     caggatgcca ttgctgtgga agctgcctgc actaatgttc cggcgttatt tcttgatgtc       480
     tctgaccaga cacccatcaa cagtattatt ttctcccatg aagacggtac gcgactgggc       540
     gtggagcatc tggtcgcatt gggtcaccag caaatcgcgc tgttagcggg cccattaagt       600
     tctgtctcgg cgcgtctgcg tctggctggc tggcataaat atctcactcg caatcaaatt       660
     cagccgatag cggaacggga aggcgactgg agtgccatgt ccggttttca acaaaccatg       720
     caaatgctga atgagggcat cgttcccact gcgatgctgg ttgccaacga tcagatggcg       780
     ctgggcgcaa tgcgcgccat taccgagtcc gggctgcgcg ttggtgcgga tatctcggta       840
     gtgggatacg acgataccga agacagctca tgttatatcc cgccgtcaac caccatcaaa       900
     caggattttc gcctgctggg gcaaaccagc gtggaccgct tgctgcaact ctctcagggc       960
     caggcggtga agggcaatca gctgttgccc gtctcactgg tgaaaagaaa aaccaccctg      1020
     gcgcccaata cgcaaaccgc ctctccccgc gcgttggccg attcattaat gcagctggca      1080
     cgacaggttt cccgactgga aagcgggcag tga                                   1113
```

Fig. 40: DNA sequence used, referred to as DNA_BASE(i).

The ***DNA*** data we used was derived from the ***E. coli laci*** gene as shown in **Fig. 40**.[10] Addressing all the above-mentioned concerns, our coding scheme ***Ω→ℜ*** was as follows. The values were mapped onto ℜ

[10] The DNA data comes from **EMBOSS** which is available online from **sourgeforce,net** at http://embossgui. sourceforge.net /demo/manual/ dottup.html.

through a polar projection defined by the vector $\boldsymbol{P^T=(A=0,T=-\pi/2,G=-\pi,C=3\pi/2)}$. Then, the signal values were amplified by an order of magnitude (***10·P***). Next, to generate observations ***y(i)*** exhibiting a *sufficiently large* set of values in $\Re$, multiple values were added as discussed shortly. Let ***DNA_BASE(i)*** be a ***unit*** row vector (***A,T,G,C***) where a base is said to be "***on"*** and thus represented by a ***one*** and remainder other three bases are said to be "***off***" and represented by a ***zero***. This way, the coding scheme $\boldsymbol{\Omega\rightarrow\Re}$ is succinctly described as follows.

$$\boldsymbol{DNA(i) = AP(i) + AP(i\text{-}1) + AP(i\text{-}2) + AP(i\text{-}3)}$$
$$\textit{where } \boldsymbol{AP=10\cdot P(i)} \textit{ and } \boldsymbol{P(i)= DNA\_BASE(i)\cdot P}. \quad (10.1)^{11}$$

Note that this coding scheme $\boldsymbol{\Omega\rightarrow\Re}$ encodes not *single* bases but rather ***4***-base sequences (e.g., ***AAAG, CAAA, TAAA, CAAT, CAAG***, etc.). As a result, representative values for unearthed localized stationary conditions map to *approximate* stable patterns of repeated (***4***-base) *motifs* hidden within the input signal. **Fig. 41** shows the sequence.

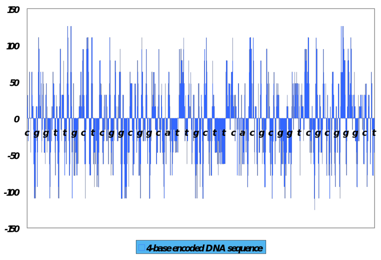


Fig. 41: Time plot of the well-conditioned DNA(i) time series.

Next, we applied the **HPS TRANSFORM** to the *well-conditioned* ***DNA***. Note that the **HPS TRANSFORM** was applied – as in *all* examples – using default values of ***α=0.005, m=30,*** and ***m'=15***. Moreover, the ***DNA*** sequence we used had only ***1113*** bases; therefore, the *original* sequence was repeated until reaching the ***N=3600*** samples required by our reference implementation.

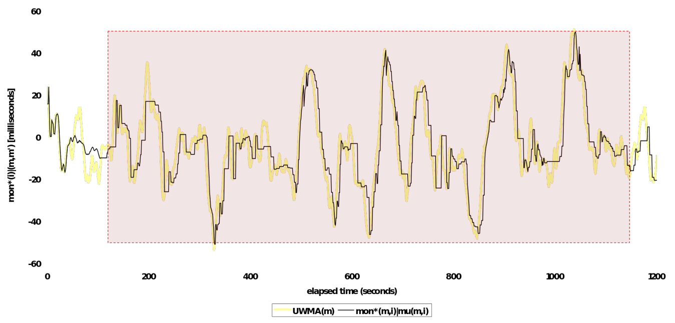
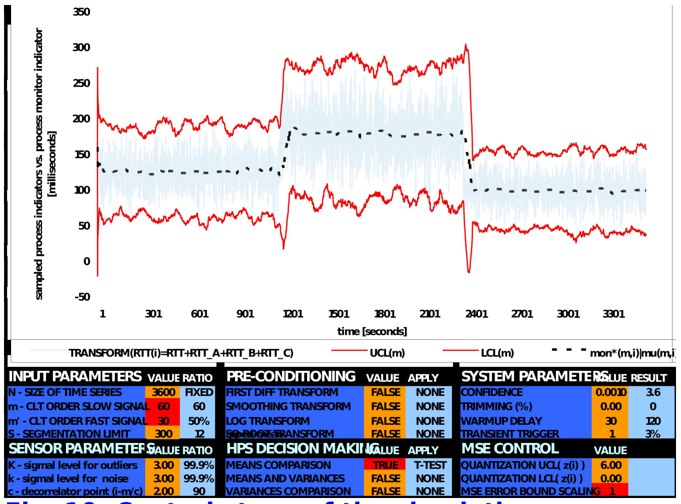


Fig. 42: HPS monitor signal for the DNA time series.

Fig. 42 shows the resultant **HPS monitor signal** plotted together with corresponding duration for unearthed **ATS segments**. With respect to HPS FRACTALITY, the resultant **HPS approximation** had ***754*** non-*trivial* **ATS segments**, which ranged in duration from ***2*** through ***35*** and were associated with an average **HPS segment duration** of ***4*** units (which represent here *overlapping* ***4***-base *motifs*). That is, the stationary-based approximation exhibited significant **HPS compressibility** (***67%***) for the ***DNA*** time series. Moreover, *w.r.t.* GOODNESS OF FIT, average ***z-value*** for the $\boldsymbol{\varepsilon_{fast}}$ residual was an *unbiased* ***0.00*** coupled with a standard deviation of *just* ***0.19***.[12] As expected, ***z-values*** (for the $\boldsymbol{\varepsilon_{fast}}$ residual) could be approximated as ***~N(0.00, 0.04)*** white noise.[13] In summary, resultant **HPS approximations** (under *default* parameters) unearthed significant amount of **approximate τ-invariance** from our (**10.1**) *well-conditioned* ***DNA*** sequence and again, error behavior for **HPS approximations** was consistent with previous findings.

SECTION 8.3: DETAILS OF THE SIMULATION

We simulated the *online* HPS TRANSFORM. This simulation was implemented using **MS EXCEL 2003**. The front-end of the simulation is shown in **Fig. 18**, which has three components: **(1)** SUMMARY MEASURES (*not shown*) for the resultant **HPS approximation**, **(2)** GRAPHICAL DISPLAY of the **HPS approximation** (*right top panel*), and **(3)** PARAMETER CONTROL (*right bottom panel*).



350
300
250
200
150
100
50
0
-50
sampled process indicators vs. process monitor indicator [milliseconds]
1
301
601
901
1201
1501
1801
2101
2401
2701
3001
3301
time [seconds]
TRANSFORM(RTT(i)=RTT+RTT_A+RTT_B+RTT_C)
UCL(m)
LCL(m)
mon*(m,i)|mu(m,i

INPUT PARAMETERS	VALUE	RATIO	PRE-CONDITIONING	VALUE	APPLY	SYSTEM PARAMETERS	VALUE	RESULT
N - SIZE OF TIME SERIES	3600	FIXED	FIRST DIFF TRANSFORM	FALSE	NONE	CONFIDENCE	0.0010	3.6
m - CLT ORDER SLOW SIGNAL	60	60	SMOOTHING TRANSFORM	FALSE	NONE	TRIMMING (%)	0.00	0
m' - CLT ORDER FAST SIGNAL	30	50%	LOG TRANSFORM	FALSE	NONE	WARMUP DELAY	30	120
S - SEGMENTATION LIMIT	300	12	SQ-ROOT TRANSFORM	FALSE	NONE	TRANSIENT TRIGGER	1	3%
SENSOR PARAMETERS	**VALUE**	**RATIO**	**HPS DECISION MAKING**	**VALUE**	**APPLY**	**MSE CONTROL**	**VALUE**	
K - sigmal level for outliers	3.00	99.9%	MEANS COMPARISON	TRUE	T-TEST	QUANTIZATION UCL(z(i))	6.00	
k - sigmal level for noise	3.00	99.9%	MEANS AND VARIANCES	FALSE	NONE	QUANTIZATION LCL(z(i))	0.00	
c - decorrelator point (i-m/c)	2.00	90	VARIANCES COMPARISON	FALSE	NONE	MSE ERROR BOUND SCALING	1	

Fig. 10: Control stage of the simulation.

The simulation was implemented on an **MS WINDOWS XP w/ SP2** ***3GHz*** Intel***4*** PC with ***512MB***. The algorithm was simulated using (**XML** data propagation over multiple) computationally intensive WORKSHEETS contained within one **MS EXCEL 2003** WORKBOOK. Complementary analysis was performed with **JMP 5.1 FOR WINDOWS**. Exploration and visualization of the parameter space was made with **XLSTAT FOR MICROSOFT EXCEL**. The time needed for the WORKBOOK to update its analyses was about ***15 seconds*** for the provided ***N=3600*** dataset. This implies a loose upper

[11] For completeness, Y(2)=AP(i)+AP(2) and Y(1)=AP(1).

[12] For reference purposes, note that the encoded ranged of the ***DNA*** time series was from ***-125*** to ***126*** as shown in **Fig. 41**.

[13] Assuming proper CLT stabilization orders, such is expected of *both* **HPS quantization errors** but not of **HPS relative error**, which retains the distribution of the original signal. For completeness, average **HPS relative error** was ***-1.37*** under a standard deviation of ***17.37***. Finally, the number of heavy-tailed outliers detected was ***9*** heavy-tail outliers.

bound of ***four milliseconds*** for each of the ***N*** iterations. Note that this number is due to a high-level simulation that is graphics-intensive and generates extensive complementary statistical analyses.[14] The size of the workbook is ***85MB*** (***16MB*** compressed). To test the simulation with the data included, just go to the worksheet named "IP" to monitor the **HPS simulation** with parameters of your choosing.[15] To test the simulation with data of your own, you will need a time series of ***N=3600*** data points.[16] Then, just paste it onto column "**XYZ**" on the worksheet named "**TS**". Now, erase the contents of the adjacent three data columns. Recalculate (F9) to update the workbook.[17] More information about the simulation is provided on **Appendix**.

[14] Note that for an online implementation (for example, in C under Linux without such visualization as typically done in a network stack implementation), this bound will be at least several orders of magnitude less, a result of its ***O(1)*** worst case running time.

[15] As a general practice, please always set "Tools – Macro – Security" option to Very High. This simulator will run in such setup, although a dialog box may remind you of such setting

[16] If somewhat less than ***3600*** data points are available, padding can be done

[17] Because of implementation, if initial values are repeated, the variance will initialize to ***zero***, which will propagate throughout as to ***DIV0!*** To prevent this error, preconditioning of the input data must be done. This is done by simply adding a very small noise ***‹d(i)›*** to ***‹y(i)›***, so that each ***‹y(i)›*** becomes ***‹y(i)›+ ‹d(i)›***). This is shown in the **DNA** example.