Large data has been accumulating in all aspects of our lives for quite some time. Advances in sensor technology, the Internet, wireless communication, and inexpensive memory have all contributed to an explosion of “Big Data”. System of Systems (SoS) integrate independently operating, non-homogeneous systems to achieve a higher goal than the sum of the parts. Today’s SoS are also contributing to the existence of unmanageable “Big Data”. Recent efforts have developed a promising approach, called “Data Analytics”, which uses statistical and computational intelligence (CI) tools such as principal component analysis (PCA), clustering, fuzzy logic, neuro-computing, evolutionary computation (such as genetic algorithms), Bayesian networks, etc. to reduce the size of “Big Data” to a manageable size and apply these tools to (a) extract information, (b) build a knowledge base using the derived data, and (c) eventually develop a non-parametric model for the “Big Data”. This paper demonstrates how to construct a bridge between SoS and Data Analytics to develop reliable models for such systems. The subject material for this demonstration is using data analytics to generate a model to forecast produced photovoltaic energy to assist in the optimization of a micro grid SoS. Tools like fuzzy interference, neural networks, PCA, and genetic algorithms are used.

1. Introduction

System of Systems (SoS) are integrated, independently operating systems working in a cooperative mode to achieve a higher performance. A detailed literature survey on definitions to applications of SoS can be found in recent texts by Jamshidi [1,2]. Application areas of SoS are vast indeed. They are software systems like the Internet, cloud computing, health care, and cyber-physical systems all the way to such hardware dominated cases like military, energy, transportation, etc. Data analytics and its statistical and intelligent tools including clustering, fuzzy logic, neuro-computing, data mining, pattern recognition and post-processing such as evolutionary computations have their own applications in forecasting, marketing, politics, and all domains of SoS.

A typical example of SoS is the future Smart Grid, destined to replace conventional electric grid. A small-scale version of this SoS is a micro-grid designed to provide electric power to a local community. A Micro-Grid is an aggregation of multiple distributed generators (DGs) such as renewable energy sources, conventional generators, in association with energy storage units which work together as a power supply networked in order to provide both electric power and thermal energy for small communities which may vary from one common building to a smart house or even a set of complicated loads.
consisting of a mixture of different structures such as buildings, factories, etc. [2]. Typically, a micro-grid operates synchronously in parallel with the main grid. However, there are cases in which a Micro-Grid operates in islanded mode, or in a disconnected state [3]. Accurate predictions of received solar power can reduce operating costs by influencing decisions regarding buying or selling power from the main grid or utilizing non-renewable energy generation sources.

The object of this paper is to use a massive amount of data on solar irradiance as an integral system of the micro-grid SoS to extract relevant information for available solar energy in an attempt to derive an unconventional model.

Section 2 first describes the micro-grid that will be used as the SoS of interest for this paper. Section 3 then describes the set of environmental data to be used in this paper as the input to the different data analytics tools. Section 4 then discusses the application and effectiveness of different data analytics tools in the generation of models and relations that could be leveraged to better optimize the operation of the micro-grid. Finally, Section 5 summarizes the work done in this paper and draws conclusions based on their findings.

2. System model

The micro-grid SoS under consideration is shown in Fig. 1. Shown here are solar array, battery storage, DC–AC inverter, load and a controller to manage the entire system. Ultimately, we want to forecast received solar power as a model based on real-time environmental measurements to be used in an energy management system [3] to minimize operating costs.

This micro-grid represents a facility scale Cyber-Physical System (CPS) or a SoS consisting of a building with:

- A fixed (or with tracking system) solar photovoltaic system.
- A load demand in the form of overall energy consumption, HVAC and lighting, with bi-directional communications (e.g. bi-directional inverter).
- A reconfigurable control and acquisition system (i.e. with open I/O modules, embedded controller for communication, processing and a user-programmable FPGA).
- A local, off-site or cloud-based computing infrastructure for simulation/computational analysis.

3. PV data description

To ensure the Photovoltaic (PV) input data for the different data analysis tools is comprehensive, data from different sources was combined to form the full dataset. This was possible because of the solar research projects occurring in Golden, CO, where the National Renewable Energy Laboratory (NREL) is conducting long term research and data recording to support the growing renewable energy industry.
The first source was the data recorded by the Solar Radiation Research Laboratory (SRRL), which employs over 70 instruments to measure solar conditions and environmental parameters [5]. Also, this data set includes 180° images of the sky that are used to determine current cloud conditions directly. Fig. 2 shows the SRRL imaging instrument, and Fig. 3 is one of the images included in the dataset.

The second source of data was the SOLPOS data, made available by the Measurement and Instrumentation Data Center (MIDC), which has stations throughout North America to capture information on solar position and available solar energy [6]. Luckily, the MIDC has a station near NREL, so their data can be used in conjunction with the SRRL data.

The final set of data originates from the Iowa Environmental Mesonet (IEM) [7]. Their Automated Surface Observing System (ASOS) station near the Golden, CO site was also included to have current weather data in the set.

Data from the month of October 2012 was combined from the different sources of data. This final set includes one sample for each minute of the month and incorporates measured values for approximately 250 different variables at each data point. The data set was sanitized to only include data points containing valid sensor data prior to the analysis.

4. Data analytics of PV data

In this section, the analysis steps are described, and the results from the different techniques are compared. This section primarily discusses the analysis of the second dataset discussed in Section 3. The goal is to use data analytics tools to generate a useful model from the dataset without needing to resort to parametric analysis and the use of subject matter experts.

4.1. Objective identification

Since the micro-grid would benefit from predicted values of solar irradiance, it was decided that the output of the data analytics should be predicted values of three key irradiance parameters (Global Horizontal Irradiance (GHI), Direct Horizontal Irradiance (DHI), and Direct Normal Irradiance (DNI)). These values were shifted by 60 min so that they would serve as output datasets for the training of the Fuzzy Inference System and Neural Network fitting tools that ultimately provided the means of non-parametric model generation in this exercise.

4.2. Input variable downselection

Once the objective of the data analytics was determined, relevant inputs to the data analytics tools needed to be identified. The full dataset contains approximately 250 different variables. Unfortunately, due to the curse of dimensionality, including all these variables in the data analytics was not practical due to memory and execution time constraints. If this exercise was to be conducted using distributed “Cloud” computing, the number of variables to be considered might not need
to be down-selected; however, since this effort took place on a single PC, the number of variables needed to be reduced. Ideally, a subject matter expert would be available to optimally identify the best variables to include in the evaluated dataset, or an adaptive training algorithm could be used to automatically perform the selection process. For the purposes of this paper, several variables were selected based on intuition, including cloud levels, humidity, temperature, wind speed, and current irradiance levels.

4.3. Cleanup of the raw dataset

Next, cleanup of the reduced dimension dataset was started by removing all data points containing invalid values from the data set. For instance, during night hours, many solar irradiance parameters contained negative values. Once these invalid data points were removed, the data set was further reduced by removing data points in which GHI, DHI, and DNI levels were very low. The primary reason for this second step was to reduce the amount of time and memory necessary for analysis. Fig. 4 contains the measurements of GHI, DHI, and DNI over one day in the cleaned dataset.

4.4. Non-parametric model generation tools

After cleaning took place, the data could be fed into either of the two non-parametric model generating tools, the Fuzzy Inference System Generator and Back-Propagation Neural Network training tools included in the Matlab Fuzzy Logic Toolbox and the Neural Network Toolbox.

Fig. 3. Sample sky image.

Fig. 4. Three key irradiance parameter plot for a clear day.
4.4.1. Non-parametric model generation tools

The Matlab Fuzzy Logic Toolbox function used in this step, genfis3 uses Fuzzy C-Means clustering to cluster values for each variable which produces fuzzy membership functions for each of the variables in the input matrix and output matrix. It then determines the rules necessary to map each of the fuzzy inputs to the outputs to best match the training data set. These membership functions and rules can be viewed using the Matlab FIS GUI tools such as ruleview. Fig. 5 shows the results of running genfis3 on only four different variables in the dataset. It shows that four clusters were generated using Fuzzy C-Means for each input and output variable.

When run with default parameters, the genfis3 function ran significantly slower and performed worse than Matlab’s Neural Network fitting function.

Note in Fig. 6, differences in the observed and predicted data points generally corresponds to the presence of clouds or other anomalies that could not be predicted an hour in advance using the variables input to the function.

4.4.2. Neural network fitting tool

The second model generating method used was the Matlab Neural Network Training tool. By default, this tool uses the Levenberg–Marquardt back propagation method to train the network to minimize its mean squared error performance. Fig. 7 shows a representation of the feed-forward neural network generated when training using 13 inputs variables and one hidden layer comprised of 10 neurons.

Results from training one sample set are shown in Figs. 8–10.

4.5. Additional pre-processing discussion

Once the initial performance of these two tools was observed and evaluated, it was decided that further effort should go into including a greater number of original input variables and including additional preprocessed parameters in the training data in an effort to enhance the performance of the derived model. This effort took three paths, the calculation of nonlinear input parameters, the inclusion of a greater number of input parameters, and the reduction of input data dimension when necessary in order to support the execution requirements of the two model generation tools.

4.5.1. Nonlinear data set expansion

In an effort to derive additional useful input parameters from the existing dataset, each variable included in the dataset generated several additional variables based on nonlinear functions and past values of the variable itself. Additional derived variables include \( x(t)^2 \), \( \sin(x(t)) \), \( \cos(x(t)) \), slope\( (x(t-1):x(t)) \), slope\( (x(t-60):x(t)) \), mean\( (x(t-60):x(t)) \), and stdev\( (x(t-60):x(t)) \). Inclusion of these parameters in the training data set greatly improved the performance of the training
tools. These functions were chosen to add to the training data information based on recent timesteps (slope and stdev). Since some of the raw data provided was angular in nature, the trigonometric functions were included to try to make the data more naturally informative to the training model. A subject matter expert would be useful in this step to identify useful derived parameters such as these to add to the training data set.

Fig. 11 contains the linear regression performance of a neural network trained including these additional variables.

![Fig. 6. Data generated using GENFIS3 based on 13 input variables.](image)

![Fig. 7. Trained feed-forward neural network representation.](image)

![Fig. 8. Back-propagation performance curve.](image)
4.5.2. Large data sets and principal component analysis (PCA)

Models were generated using different sets of input variables to try to assess the impact of incorporating increasing numbers of variables in the training data set. In general, the trained model performed better when more variables were included in the training data set; however, as the number of variables increased, the training execution time became excessive and out-of-memory errors occurred when the data sets became too large.

In order to combat this issue, the dimension of the training data set was reduced to a manageable size using PCA. PCA can be used to compress the information from a large number of variables to a smaller dataset while minimizing the information lost during this process [8,9]. This can be performed directly on a dataset using the `princomp` function in Matlab.

The columns of the SCORE matrix returned by the `princomp` function represent the columns of the input data transformed to place the majority of the information in the data set in the first few principal components. The information distribution among the principal components is illustrated in Fig. 12. The higher eigenvalues represent the principal components with the most information. Incorporating principal components past 10 provides minimal additional information.

Fig. 13 below shows the quality of information recovery if transforming back to the original basis using only information from the first 50 principal components.

In this application PCA was primarily useful because it allowed the reduction of very high dimension data sets to smaller, more manageable data sets that could be used as training inputs to the model generation tools.

![Fig. 9. Post training network regression performance.](image1)

![Fig. 10. Data generated using NFTOOL based on 13 input variables and 10 hidden neurons.](image2)
Fig. 11. Post training network regression performance, incorporated additional nonlinear parameters.

Fig. 12. Principal component information graph.

Fig. 13. Data recovery demonstration using first 50 principal components.
4.5.3. Dimensional reduction using genetic algorithms

In addition to PCA, Genetic Algorithms were used to try to optimally reduce the dimension of the dataset. In a real-world application, reducing the number of required sensors equates to reduced system realization cost and model training time. This was accomplished using the Matlab Global Optimization Toolbox GA tool. This tool, when set to use the genetic algorithm solver, generates a test population to be evaluated against a defined performance function. Whichever population elements perform the best (have the smallest performance function value) will be most represented in the next generation of test elements. In general, genetic algorithms are a technique used to generate parameters, an algorithm, set of symbols, or instructions to best solve a problem. The process undertaken by this technique mimics the evolutionary adaptation of organisms to their environment as described by Charles Darwin. This is accomplished through an iterative training process, where each generation, or set of potential algorithms, is tested to determine their relative performance. Then, the top performing candidate algorithms are selected to seed the next generation of algorithms. This process is summarized in Fig. 14 below.

For this application, it was decided that data reduction would be performed using Genetic Algorithms first on the full set of initial data. Next, the remaining set of data would be non-linearly expanded. Finally, the expanded set of data would be used to train a neural network to gauge the overall effectiveness of the down selected sensors. This process is illustrated in Fig. 15.

Using the optimtool GUI (seen in Fig. 16 below), the problem to be solved was defined to have the proper number of inputs (244 for the first step, which matches the data set dimension after preprocessing/cleanup). The population type was defined to be a bit stream. Next, the handle to the performance function was entered. For this analysis, the performance function was set up to use the population element bit stream to eliminate unselected data dimensions. Then, a 10 neuron neural network was trained using this data. Next, the population per generation was set to 100, and the mutation rate was set to 10%. Finally, the performance value was calculated to be 90% based on the neural network MSE component and 10% on the training time improvement (which also takes into account the dimension of the data).

Due to time and computational restrictions, the training data set was selected to be the first 1000 preprocessed samples, although the performance of the trained neural network was checked against the entire sample set.

The first iteration of the Genetic Algorithm process took the initial 244 dimensions and reduced it to 74. Before the dimensional reduction, the total MSE of the trained neural network was approximately 2.9E5. After the dimensional reduction, the total MSE was actually reduced to approximately 1.54E5. This means that the GA algorithm reduced the number of attributes required for the network by almost 70% while reducing the network’s MSE by about 47%.

![Genetic algorithm high level description](image-url)
The next step expanded the dimension of the data by adding back the time dimension (if necessary) and then performing nonlinear expansion as described in Section 3.5.1. This expanded the dimension to 593. Using this data, a 10 neuron neural network was trained (without using PCA) that had a total MSE of approximately 3.52E3. This actually beats the other trained neural networks described in the next section.
5. Results

In order to generate the best non-parametric model possible, different combinations of data inputs to the GENFIS3 and NFTOOL were considered. Different implementations of the options discussed above were evaluated during this analysis.

The best performing NFTOOL generated model used all 244 original variables, which were then expanded the dimension to 1945 using the nonlinear variable derivation calculations. Next, the dimension of the data was shrunk to 150 so that the training function had sufficient memory to train the network. The resulting network was the best of all the generated models.

The best performing GENFIS3 generated model evaluated during this effort used the same input data set as mentioned in the paragraph above with the exception that the dimension was shrunk down to 50 using PCA. It was observed during this effort that effectiveness of the GENFIS3 tool appears to be less tolerant of high dimension training data sets than the NFTOOL.

The performance of this network is shown in Figs. 17–20.

Tables 1 and 2 describe the performance of the models generated using these tools. Note that these performance numbers should be compared qualitatively since the different input parameter configurations can yield different numbers of training data points. The green highlighted rows correspond to the model generation configurations that yielded the best R error metric values.

A sub-optimal predictor was constructed in order to show its performance relative to that of the non-parametric models. This predictor was based on the average GHI, DHI, and DNI values for each time bin in the data set. To clarify, these "Time Bin
Mean” moving window average values were used as simplistic prediction of the irradiance parameters. Table 3 shows the improvement of the non-parametric models when compared to this sub-optimal predictor.

During this analysis, the aspect of the scalability of the GENFIS3 and NFTOOL tools was evaluated. The model generation time for NFTOOL was always shorter than GENFIS3 for the same data sets. The relationship of NFTOOL execution time to dataset length and dimension was generally linear for the test cases evaluated. The relationship of GENFIS3 execution time to dataset length was also linear; however, its relationship between dataset dimension and execution time was a function of the dataset dimension squared. This is shown in Figs. 21 and 22.

The overall best trained neural network came from using the dataset generated by reducing the dimension of the initial set of 244 variables using a Genetic Algorithm. After expanding the reduced dataset using Nonlinear Expansion, the final

![Fig. 19. Best neural network DHI error.](image1)

![Fig. 20. Best neural network DNI error.](image2)

<table>
<thead>
<tr>
<th>Model type</th>
<th>Input Params</th>
<th>Add Nonlin Params</th>
<th>PCA?</th>
<th>Final Dim</th>
<th>MSE GHI</th>
<th>MSE DHI</th>
<th>MSE DNI</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIS 13</td>
<td>N</td>
<td>N</td>
<td>13</td>
<td>8.81E+03</td>
<td>4.98E+04</td>
<td>2.03E+03</td>
<td>0.906</td>
<td></td>
</tr>
<tr>
<td>FIS 32</td>
<td>N</td>
<td>N</td>
<td>32</td>
<td>8.68E+03</td>
<td>4.90E+04</td>
<td>1.99E+03</td>
<td>0.907</td>
<td></td>
</tr>
<tr>
<td>FIS 32</td>
<td>Y</td>
<td>Y</td>
<td>10</td>
<td>6.64E+03</td>
<td>3.58E+04</td>
<td>1.70E+03</td>
<td>0.938</td>
<td></td>
</tr>
<tr>
<td>FIS 32</td>
<td>Y</td>
<td>Y</td>
<td>50</td>
<td>6.35E+03</td>
<td>3.12E+04</td>
<td>1.85E+03</td>
<td>0.945</td>
<td></td>
</tr>
<tr>
<td>FIS 13</td>
<td>Y</td>
<td>N</td>
<td>97</td>
<td>4.13E+03</td>
<td>2.31E+04</td>
<td>1.14E+03</td>
<td>0.961</td>
<td></td>
</tr>
<tr>
<td>FIS 244</td>
<td>N</td>
<td>Y</td>
<td>50</td>
<td>1.06E+04</td>
<td>5.00E+04</td>
<td>2.40E+03</td>
<td>0.903</td>
<td></td>
</tr>
<tr>
<td>FIS 244</td>
<td>Y</td>
<td>Y</td>
<td>50</td>
<td>2.96E+03</td>
<td>2.08E+04</td>
<td>1.05E+03</td>
<td>0.967</td>
<td></td>
</tr>
<tr>
<td>FIS 244</td>
<td>Y</td>
<td>Y</td>
<td>150</td>
<td>1.82E+04</td>
<td>5.50E+04</td>
<td>3.35E+03</td>
<td>0.901</td>
<td></td>
</tr>
</tbody>
</table>
Table 2
Performance comparison of the generated non-parametric models (NN10).

<table>
<thead>
<tr>
<th>Model type</th>
<th>Input Params</th>
<th>Add Nonlin Params</th>
<th>PCA?</th>
<th>Final Dim</th>
<th>MSE GHI</th>
<th>MSE DHI</th>
<th>MSE DNI</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN10</td>
<td>13</td>
<td>N</td>
<td>N</td>
<td>13</td>
<td>7.01E+03</td>
<td>3.41E+04</td>
<td>1.65E+03</td>
<td>0.934</td>
</tr>
<tr>
<td>NN10</td>
<td>32</td>
<td>N</td>
<td>N</td>
<td>32</td>
<td>6.94E+03</td>
<td>3.40E+04</td>
<td>1.96E+03</td>
<td>0.934</td>
</tr>
<tr>
<td>NN10</td>
<td>32</td>
<td>Y</td>
<td>Y</td>
<td>249</td>
<td>1.22E+03</td>
<td>3.83E+03</td>
<td>5.28E+02</td>
<td>0.992</td>
</tr>
<tr>
<td>NN10</td>
<td>32</td>
<td>Y</td>
<td>Y</td>
<td>10</td>
<td>4.17E+03</td>
<td>1.74E+04</td>
<td>1.14E+03</td>
<td>0.969</td>
</tr>
<tr>
<td>NN10</td>
<td>32</td>
<td>Y</td>
<td>Y</td>
<td>50</td>
<td>2.21E+03</td>
<td>7.14E+03</td>
<td>9.74E+02</td>
<td>0.986</td>
</tr>
<tr>
<td>NN10</td>
<td>13</td>
<td>Y</td>
<td>N</td>
<td>97</td>
<td>1.72E+03</td>
<td>4.57E+03</td>
<td>6.58E+02</td>
<td>0.991</td>
</tr>
<tr>
<td>NN10</td>
<td>32</td>
<td>Y</td>
<td>N</td>
<td>249</td>
<td>1.20E+03</td>
<td>3.52E+03</td>
<td>4.20E+02</td>
<td>0.993</td>
</tr>
<tr>
<td>NN10</td>
<td>244</td>
<td>N</td>
<td>Y</td>
<td>50</td>
<td>5.26E+03</td>
<td>1.63E+04</td>
<td>1.57E+03</td>
<td>0.966</td>
</tr>
<tr>
<td>NN10</td>
<td>244</td>
<td>Y</td>
<td>Y</td>
<td>150</td>
<td>9.57E+02</td>
<td>2.95E+03</td>
<td>4.57E+02</td>
<td>0.994</td>
</tr>
</tbody>
</table>

Table 3
Performance of best non-parametric to mean time bin sub-optimal predictor.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Model type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time bin mean</td>
<td>Best FIS</td>
</tr>
<tr>
<td>MSE GHI</td>
<td>1.17E+04</td>
</tr>
<tr>
<td>% MSE GHI improvement</td>
<td>0%</td>
</tr>
<tr>
<td>MSE DHI</td>
<td>8.44E+04</td>
</tr>
<tr>
<td>% MSE DHI improvement</td>
<td>0%</td>
</tr>
<tr>
<td>MSE DNI</td>
<td>4.51E+03</td>
</tr>
<tr>
<td>% MSE DNI improvement</td>
<td>0%</td>
</tr>
<tr>
<td>R</td>
<td>8.39E-01</td>
</tr>
<tr>
<td>% R improvement</td>
<td>100%</td>
</tr>
</tbody>
</table>

Fig. 21. Model generation execution time relationship with dataset dimension.

Fig. 22. Model generation execution time relationship with dataset length.
trained data set had a MSE GHI of 7.42E + 02, a MSE DNI of 2.60E + 03, and a MSE GHI of 1.78E + 02. Note that this solution is also preferable because it would require less sensors to implement.

6. Conclusions

This paper presents a high level look at some of the tools available in the Matlab toolset that enable the user to extract information from “Big Data” sources in order to draw useful conclusions. As described in Section 2, the specific application discussed in this paper is the prediction of the amount of solar power generated by a micro-grid. Section 3 then discusses the data that was gathered to support this exercise. Section 4 discusses the steps and techniques considered while trying to generate the best solar irradiance prediction model. Techniques discussed included dataset sanitation, training input parameter selection, model generation via Fuzzy C-Means Clustering and Rule Inference (GENFIS3), Neural Network training using back propagation (NFTOOL), Pre-Processing nonlinear variables to add to the training data set, the use of PCA to reduce the dimension of the training data while maximizing the information retained in the data set, and using Genetic Algorithms to optimize the reduction of trained network error and the dimension of the input dataset.

It was observed in the results presented in Section IV that the best model predicting solar irradiance was one utilizing the maximum number of original and preprocessed variables, which was then reduced to a manageable dimension using PCA prior to use in training the model. The results in this section also showed that the non-parametric model generation methods discussed in this paper performed significantly better than a sub-optimal predictor. Finally, the results describing the model generation times for the two techniques showed that NFTOOL provides significantly better training times, especially when the dimension of the dataset is high.

Future work on this topic is planned to address the benefits of the use of cloud computing to generate models for larger data sets, and the design and evaluation of a controller to buy or sell power from the grid based on demand and predictions of received solar energy.

References


Barnabas Tannahill graduated from The University of Texas at Austin with a B.S. degree in Electrical Engineering in 2005. He is currently studying at The University of Texas at San Antonio working towards a M.S. degree in Electrical Engineering while working full-time at Southwest Research Institute (San Antonio, TX).

Mo M. Jamshidi (Fellow IEEE, Fellow ASME, A. Fellow-AIAA, Fellow AAAS, Fellow TWAS, Fellow NYAS) received BS in EE, Oregon State University, Corvallis, OR, USA in 1967, the MS and Ph.D. degrees in EE from the University of Illinois at Urbana-Champaign, IL, USA in June 1969 and February 1971, respectively. He holds honorary doctorate degrees from University of Waterloo, Canada, 2004 and Technical University of Crete, Greece, 2004. Currently, he is the Lutcher Brown Endowed Chaired Professor at the University of Texas, San Antonio, TX, USA. He has been an advisor to NASA (including 1st MARS Mission), USAF, USDOE and EC/EU (Brussels). He has over 680 technical publications including 68 books (12 text books), research volumes, and edited volumes in English and a few foreign languages. He is the Founding Editor or co-founding editor or Editor-in-Chief of 5 journals including IEEE Control Systems Magazine and the IEEE Systems Journal. He is an Honorary Professor at three Chinese Universities (Nanjing and Xi’an), Deakin University (Australia), Birmingham University (UK), and Obuda University (Hungary). In October 2005 he was awarded the IEEE’s Norbert Weiner Research Achievement Award. He is a member of the University of the Texas System Chancellor’s Council since 2011. He is currently involved in research on System of Systems engineering with emphasis on cloud computing, robotics, UAVs, biological and sustainable energy systems.