DISTRIBUTED AND ADAPTIVE PERSONALIZED ARTIFICIAL PANCREAS

PROJECT DESCRIPTION

Diabetes mellitus is a disease of chronically elevated blood glucose concentrations called hyperglycemia. Blood glucose is an important source of energy for our body and comes from the carbohydrate content of foods we eat. Its significance is also illustrated by the fact that the mammalian brain depends on glucose as its main source of energy. Due to this, blood glucose levels are tightly regulated to be maintained within a narrow range, even though our body is constantly challenged by perturbations such as food intake, exercise, and stress. This phenomenon of tight regulation is referred to as glucose homeostasis and achieved by negative feedback mediated by mainly two hormones: insulin and glucagon. During digestion, carbohydrates are broken down to glucose molecules. These molecules are absorbed into the bloodstream across the mucosa of the small intestine, causing a rise in blood glucose levels. This increase signals the pancreas to release insulin into the bloodstream. Insulin helps cells take in glucose from the bloodstream, thus lowering blood glucose levels. When insulin is not appropriately available, elevated blood glucose levels cannot be decreased to restore glucose homeostasis. Glucagon counterbalances the actions of insulin. Around four to six hours after meals, the glucose levels may start decreasing below desired concentration, which is called hypoglycemia, triggering the release of glucagon from the pancreas into the bloodstream. This hormone signals the liver and muscle cells to break down their glycogen (glucose storage) into glucose. The glucose molecules are then released into the bloodstream, thus raising blood glucose levels and restoring glucose homeostasis.

There are mainly two types of diabetes: type 1 and type 2. Type 1 is caused by an autoimmune destruction of the insulin-producing β cells in the pancreas causing “absolute” insulin deficiency. On the other hand, type 2 is a result of both impaired insulin secretion and resistance to its action called “relative” insulin deficiency. The serious complications and high prevalence makes diabetes a major public health problem. The short-term complications of diabetes include ketoacidosis and hyperglycemic hyperosmolar state that can lead to even death. Prolonged exposure to elevated blood glucose can damage nerves and blood vessels, causing long-term complications such as chronic kidney disease, cardiovascular disease, lower extremity amputation, eye damage (retinopathy), and death. Diabetes is the most common cause of blindness in adults and the most common single cause of end-stage kidney failure. One of the main goals of diabetes treatment is to keep blood glucose concentrations at normal or near-normal levels to prevent these complications. In 2015, 30.3 million Americans (or 9.4 % of the population) had diabetes, which includes 1.25 million type 1 children and adults, along with 1.5 million Americans that are newly diagnosed with the disease annually**.**

Treatment for diabetes requires monitoring of blood glucose levels and a combination of lifestyle adjustments and medications. The primary treatment in all patients with type 1 diabetes is insulin in order to cover for the patient’s insulin deficiency. Although type 2 diabetes may also require insulin, the first-line treatment for type 2 diabetes is diet, weight control, physical activity, and non-insulin medications such as metformin. Two technological advances changed how insulin was used in the 1980s: 1) self-monitoring of blood glucose and 2) the use of hemoglobin A1c. Being able to monitor blood glucose any time anywhere without direct supervision by a health professional was invaluable for patients to adjust their insulin therapy 24 hours a day. In addition, the hemoglobin A1c value, which indicated the average blood glucose level over the past 2 to 3 months, showed how well diabetes was managed during the period. However, those trying to achieve better blood glucose control through intensive insulin therapy faced an increased risk of hypoglycemia (abnormally low blood glucose concentration) with severe consequences to heart and brain. Night time hypoglycemia (nocturnal hypoglycemia) was especially concerning because it often occurred without any symptoms and led to unconsciousness or even death. These problems occurred because it was exceedingly difficult for an individual to accurately predict how much insulin should be injected in any particular situation because insulin needs were affected by a myriad of factors, including food intake, exercise, illness, emotional stress, sleep deprivation, and menstrual cycle. Another challenge was that the time delay between subcutaneous insulin injection and insulin action to decrease blood glucose levels could be substantial (more than an hour in some cases). Furthermore, patients often calculated the amount of insulin by themselves which was error-prone. Maintaining optimal blood glucose control while reducing the risk of hypoglycemia became a main thrust for the development of new insulin delivery strategies. In recent years, automation of insulin delivery (also known as the artificial pancreas) has been explored as a promising technology to achieve this goal.

Artificial pancreas

The artificial pancreas has typically three parts: 1) continuous subcutaneous glucose sensor, 2) computer, and 3) continuous subcutaneous insulin pump. The computer receives glucose levels at regular intervals (e.g., 5 minutes) from the sensor and computes the amount of insulin to be administered (or withheld) *via* the pump. The adequacy of the sensor accuracy and the consideration of the time delay between subcutaneous glucose readings and actual blood concentrations are critical. Like a healthy pancreas, an insulin pump can provide both basal and bolus insulin. Basal insulin is the background insulin continuously released in small amounts throughout the day. In contrast, bolus insulin refers to the extra large amount of insulin released for a short period of time in response to food intake. As described earlier, since there is a time delay between subcutaneous insulin injection and insulin action, bolus insulin is typically given from 10 to 20 minutes prior to a meal. In September 2016, the U.S. Food and Drug Administration (FDA) approved first artificial pancreas (the MiniMed 670G system from Medtronic) that automatically monitors glucose and provides appropriate basal insulin in people 14 years of age and older with type 1 diabetes. The basal rate is automated using a proportional-integral-derivative (PID) control algorithm with insulin-on-board (IOB) feedback (insulin-on-board is the calculation that tells patients how much insulin is still active from previous bolus doses). This artificial pancreas system is not fully automated because it still depends on the user to enter meal and exercise information for bolus insulin. Although the system is a historic achievement, it has been criticized for the lack of customizability because automating insulin delivery should take into account the wide variations between people and a myriad of additional factors such as stress, sleep, medication, etc. Furthermore, to counteract the effect of insulin, a multi-hormone (e.g., insulin + glucagon) approach needs to be considered. Multiple clinical studies have been recently announced to test more advanced artificial pancreas systems. However, these approaches often focus on very specific options and are not designed to readily adapt to variations and on-going changes. As a result, proactive patients are currently building their own Do-It-Yourself artificial pancreas (the Open Artificial Pancreas Project) to achieve desired customization, raising serious safety and security concerns.

Negative feedback control

In control theory, it is known that inhibitory or negative feedback can make a system resilient towards disturbances; when designed properly, a negative feedback loop can mitigate the impact of a disturbance (unwanted input) on the output of a system. **Figure 1A** shows the output of an open-loop system contains contributions from both a desired input and unwanted disturbance. The effect of a disturbance on the output can be reduced by adding a controller and a negative feedback loop that negatively feeds the output signal back to the controller, thus suppressing it (**Figure 1B**). Intuitively, when a disturbance alters the output level, negative feedback modifies the controller action and tries to correct the output. For example, if a disturbance raises the output level, the strength of negative feedback will increase and the controller is more suppressed, decreasing the output (**Figure 1C**).

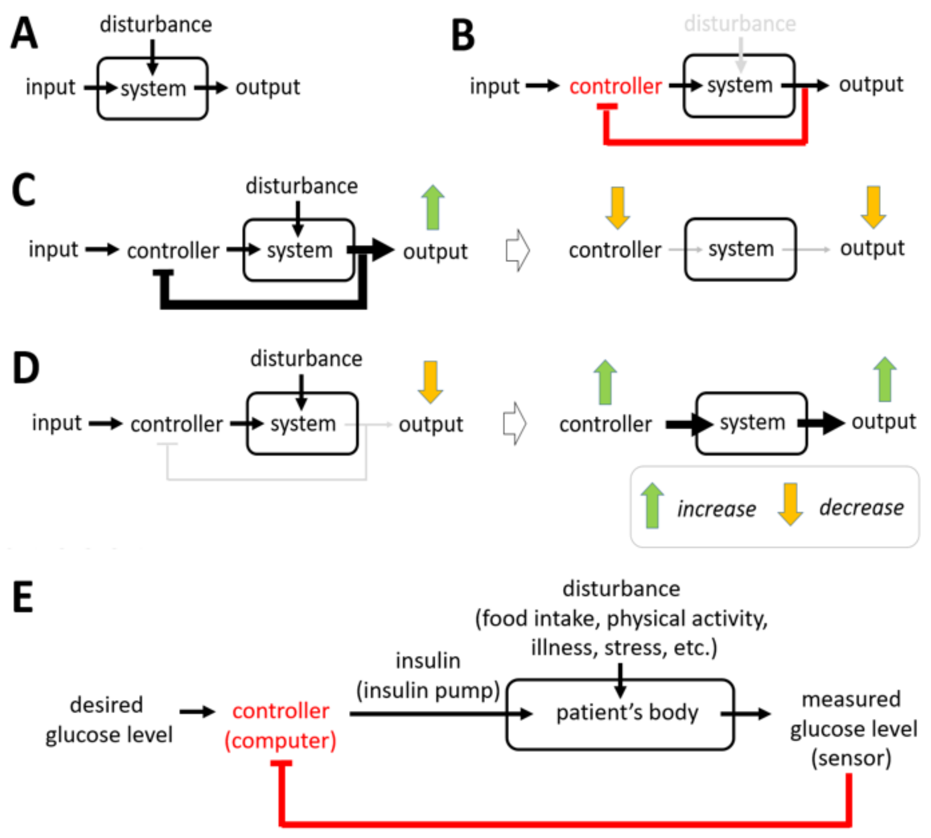
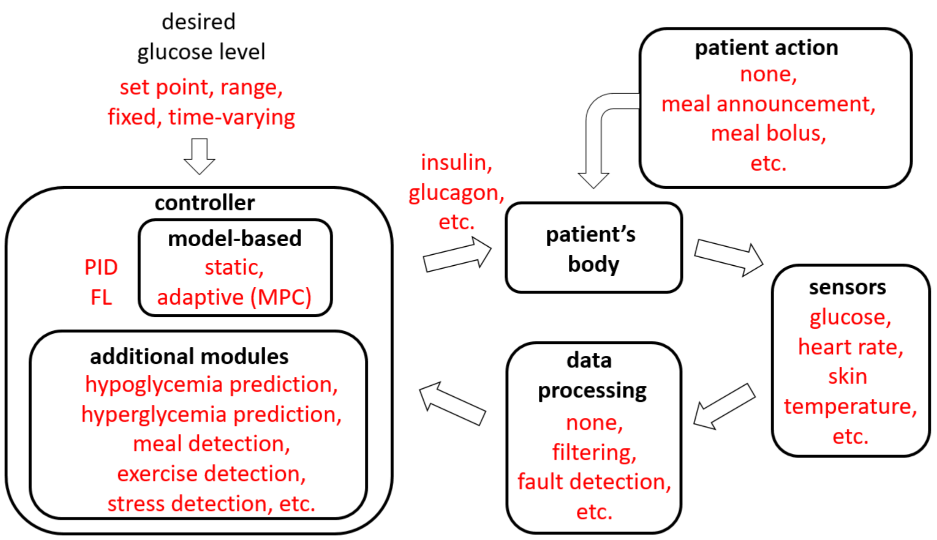


Figure 1. Artificial pancreas. (A) Open-loop control. (B-D) Closed-loop control. (E) Schematic diagram.

On the other hand, if a disturbance lowers the output, the strength of negative feedback decreases and the controller is less suppressed, raising the output (**Figure 1D**). In this fashion, the output signals can be made to follow only desired input (reference) signals but not unwanted disturbance signals. **Figure 1E** illustrates how this negative feedback control can be applied to the artificial pancreas. The input or reference signal is a desired glucose level. The controller or the control algorithm running on a computer receives the glucose sensor data, finds the difference between the desired level and the measured level, and computes the right amount of insulin to be released into the patient's body. If this negative feedback is well designed and functions properly, glucose homeostasis can be achieved in the presence of disturbances such as food intake, physical activity, illness, stress, etc.

Control objectives for the artificial pancreas

The control objectives may include maximization of time spent within a desired glucose range, minimization of hypoglycemic events, prevention of postprandial hyperglycemia, and minimization of patient intervention. These objectives need to be tailored to individual patients. For example, children are considered a high-risk group and their artificial pancreas should satisfy more stringent safety requirements. Given a set of control objectives, a suitable controller can be designed. Almost all of the possible design approaches can be represented by **Figure 2**. For each component in the figure, there are multiple design options. The first option is the desired glucose level that can be either a specific value or a range. The desired glucose level can also be fixed or change over time (time-varying). The controller receives the measured glucose level and computes the amount of insulin to be injected after comparing it to the desired concentration. Several control algorithms have been tested, including proportional-integral-derivative (PID) control, model predictive control (MPC), and fuzzy logic control (FL). These algorithms may also calculate glucagon and/or other medications as mentioned earlier and additional modules can be added to the controller, including hypoglycemia prediction, hyperglycemia prediction, meal detection, exercise detection, stress detection, etc. Model predictive control is a model-based control approach that uses adaptive system identification which will be discussed in more detail later. While proportional-integral-derivative control is simpler and widely used in science and engineering, model predictive control is considered to be more suitable for advanced multivariable artificial pancreas that uses not only glucose sensor data but also other physiological information such as heart rate and skin temperature at the expense of increased computation. These sensor data can be processed by filters and/or fault detection algorithms to increase the quality of the data before being sent to the controller. The patient may be involved in this control loop. For example, the patient may be required to make a meal announcement so that the controller can deliver appropriate bolus insulin as in the case of the first FDA-approved artificial pancreas. However, whenever a human is involved, safety concern is increased because of the unpredictable nature of human behavior.



desired glucose level
set point, range, fixed, time-varying
patient action
none, meal announcement, meal bolus, etc.
insulin, glucagon, etc.
controller
model-based
static, adaptive (MPC)
PID
FL
patient's body
sensors
glucose, heart rate, skin temperature, etc.
additional modules
hypoglycemia prediction, hyperglycemia prediction, meal detection, exercise detection, stress detection, etc.
data processing
none, filtering, fault detection, etc.

Figure 2. Possible design options for the artificial pancreas.

Multivariable system identification

To control a system effectively, the system needs to be first identified or estimated as accurately as possible. This is similar to the fact that accurate medical diagnosis (identification) should be made before appropriate treatment (control) can be done. System identification is a widely used approach in science and engineering to build mathematical models to be estimated from measured input and output data. Perfectly identifying any physical system is not feasible due to 1) system, 2) environmental, and 3) measurement uncertainties. Therefore, system identification attempts to estimate the coefficients of the system model in an optimal way. The mathematics of system identification can also be found in evidence-based medicine (EBM) that emphasizes the use of statistical evidence to improve the quality of medical care. Among EBM research designs such as cohort studies, case-control studies, cross-sectional studies, etc., randomized controlled trials, whereby subjects are randomly assigned to a target or control group, are considered gold standard. One of the most commonly used system identification approaches for randomized controlled trials is multiple linear regression. Multiple linear regression assumes the linear relationship between one dependent (output) variable and multiple independent (input) variables. In the case of diabetes, this multivariable system identification approach can be used to analyze the effect of *n* multiple physiologic variables (independent continuous random variables) x_1,…, x_n, such as heart rate and skin temperature, on glucose level y (a continuous dependent random variable) using the data y, x_1, …, x_n from *k* patients. These relationships can be collectively expressed as:

$$\begin{aligned} y_1 &= \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + \cdots + \beta_n x_{1n} + v_1 \\ &\vdots \\ y_k &= \beta_0 + \beta_1 x_{k1} + \beta_2 x_{k2} + \cdots + \beta_n x_{kn} + v_k \end{aligned} \quad [1]$$

where v_s $(1 \le s \le k, i = \text{integer})$ is a continuous random variable representing the measurement noise. $\beta_0, \dots, \beta_n$are the coefficients to be estimated. β_0 is the estimated average value of y when x_1, …, x_n are all zeros and β_t $(1 \le t \le n, j = \text{integer})$ is the estimated change in the average value of y as a result of an one-unit change in x_t $(1 \le t \le n, j = \text{integer})$. Estimated $\beta_1, \dots, \beta_n$ values show how y is correlated with x_1, …, x_n, respectively. A larger β value, either positive or negative, indicates stronger correlation. The set of equations above can be compactly put into matrix form as (vectors and matrices are shown in bold):

$$\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{v} \quad [2]$$

where:

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} \quad \boldsymbol{X} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1n} \\ 1 & x_{21} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{k1} & \cdots & x_{kn} \end{bmatrix} \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \end{bmatrix} \quad \boldsymbol{v} = \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_n \end{bmatrix}$$

By denoting estimated $\boldsymbol{\beta}$ as $\hat{\boldsymbol{\beta}}$, estimated $\boldsymbol{y}$ or $\hat{\boldsymbol{y}}$ can be written as:

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}} \quad [3]$$

The estimation error vector between $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ can be expressed as:

$$\boldsymbol{e} = \boldsymbol{y} - \hat{\boldsymbol{y}} = \boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}} \quad [4]$$

Given measured data $\boldsymbol{y}$ and $\boldsymbol{X}$, $\hat{\boldsymbol{\beta}}$ which minimizes $\|\boldsymbol{e}\|^2$ can be estimated using the least squares criterion:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \quad [5]$$

Although widely used in medical research, this multiple linear regression approach faces following issues:

1. Coefficient estimates ($\hat{\boldsymbol{\beta}}$) are "one-size-fits-all" values derived from a group of heterogeneous patients. This population-average approach may work poorly for some individuals.
2. Measured data ($\boldsymbol{y}$ and $\boldsymbol{X}$) and coefficient estimates ($\hat{\boldsymbol{\beta}}$) are assumed to be fixed or time-invariant. However, they may dynamically change over time (e.g., y_1 at time t_1 might be different from y_1 at time t_2) and these changes need to be tracked.
3. The relationship between dependent (y) and independent (x_1, …, x_n) variables is assumed to be linear although it can be non-linear in practice.
4. The approach assumes 1) statistical independence (no correlation) of the errors ($\boldsymbol{e}$), 2) normality of the error distribution, and 3) constant variance of the errors, which may not hold in practice.
5. A confounder is a variable not considered in the study but its presence affects the actual relationship between dependent and independent variables. Any confounder should be added to the study as an independent variable to remove its confounding effect. When this is not possible, confounding can be controlled through randomization, restriction, matching, etc.

Adaptive multivariable system identification for personalized artificial pancreas

High-throughput streaming data are increasingly available thanks to advances in technologies such as continuous glucose monitoring, enabling real-time system identification geared towards personalized artificial pancreas. In this section, I will develop an adaptive multivariable system identification approach that uses the real-time data (e.g., glucose concentration, heart rate, skin temperature) streaming from a single patient which adds adaptive capability to the previous system identification approach. Considering the fact that any current state is the result of previous states, it can be assumed that the current value of y depends on M previous values of y, N_1 previous values of $x_1, \dots,$ and N_n previous values of x_n:

$$y[i] = \beta_0[i] + \sum_{j=1}^{M} \alpha_j[i] y[i-j] + \sum_{j=1}^{N_1} \beta_{1,j}[i] x_1[i-j] + \cdots + \sum_{j=1}^{N_n} \beta_{n,j}[i] x_n[i-j] + v[i] \quad [6]$$

where i is the discrete-time index (integer), $\alpha_1[i], \dots, \alpha_M[i]$ are the time-varying coefficients for M previous y terms and $\beta_{n,1}[i]$, …, $\beta_{n,N_n}[i]$ are the time-varying coefficients for N_n previous $x_n$terms. $v[i]$ is a continuous random variable representing the measurement noise. Measured data $y[i], \ldots, y[i-M], x_1[i-1], \ldots, x_1[i-N_1], \ldots, x_n[i-1], \ldots, x_n[i-N_n]$ and all β coefficients are no longer fixed and can change over time. In addition, the glucose concentration y now shows up on both sides of Eq. 6 as a dependent ($y[i]$) and independent ($y[i-1], \ldots, y[i-M]$) variables, indicating that current glucose level can depend on previous glucose levels as well. Eq. 6 can be compactly put into matrix form as:

$$y[i] = \boldsymbol{u}[i]^T \boldsymbol{w}[i] + v[i] \qquad [7]$$

where:

$$\boldsymbol{u}[i] = \left[1, y[i-1], \cdots, y[i-M], x_1[i-1], \cdots, x_1[i-N_1], \ldots, x_n[i-1], \cdots, x_n[i-N_n]\right]^T$$

$$\boldsymbol{w}[i] = \left[\beta_0[i], \alpha_1[i], \cdots, \alpha_M[i], \beta_{1,1}[i], \cdots, \beta_{1,N_1}[i], \cdots, \beta_{n,1}[i], \cdots, \beta_{n,N_n}[i]\right]^T \qquad [8]$$

By denoting estimated $\boldsymbol{w}$ as $\hat{\boldsymbol{w}}$, estimated $y[i]$ or $\hat{y}[i]$ can be expressed as:

$$\hat{y}[i] = \boldsymbol{u}[i]^T \hat{\boldsymbol{w}}[i] \qquad [9]$$

The estimation error or the difference between $y[i]$ and $\hat{y}[i]$ can be expressed as:

$$e[i] = y[i] - \hat{y}[i] = y[i] - \boldsymbol{u}[i]^T \hat{\boldsymbol{w}}[i] \qquad [10]$$

Given $y[i]$ and $\boldsymbol{u}[i]$ which are measured data**,** $\hat{\boldsymbol{w}}[i+1]$ can be estimated by minimizing the expected value of $e[i]^2$ or $E\{e[i]^2\}$. Using the stochastic-gradient recursion, $E\{e[i]^2\}$ can be approximated and a recursive relation for updating $\hat{\boldsymbol{w}}[i]$ can be shown as:

$$\hat{\boldsymbol{w}}[i+1] = \hat{\boldsymbol{w}}[i] + \mu \boldsymbol{u}[i] e[i] \qquad [11]$$

where μ is a controllable step-size parameter. There is a trade-off between the estimation error and coefficient convergence rate and μ is often tuned to achieve desired performance**.** Eq. 11 is the Least Mean Squares (LMS) algorithm, one of adaptive system identification techniques used in science and engineering to identify dynamic physical systems (e.g., Unmanned Aerial Vehicle (UAV), Global Positioning System (GPS), etc.) in the presence of uncertainties. Although simple in implementation, LMS is capable of delivering high performance through iterative adaptation or "learning" over time. I previously demonstrated that the dynamic behavior of the p53-Mdm2 gene network in individual cells can be tracked using adaptive system identification algorithms (normalized LMS, RLS (Recursive Least Squares), and the Kalman filter) and the resulting time-varying models were able to approximate experimental measurements more accurately than time-invariant models. These adaptive methods can be considered as variants of recurrent neural networks studied in deep learning which exhibit dynamic temporal behavior. The outcome of Eq. 11 can also be used to predict future y value $\hat{y}[i+1]$ (see Eq. 9):

$$\hat{y}[i+1] = \boldsymbol{u}[i+1]^T \hat{\boldsymbol{w}}[i+1] \qquad [12]$$

where $\boldsymbol{u}[i+1]$ includes only the data measured before $i+1$ (see Eq. 8) and $\hat{\boldsymbol{w}}[i+1]$ is computed using $\hat{\boldsymbol{w}}[i]$ and previous data ($y[i]$ and $\boldsymbol{u}[i]$) (see Eqs. 10 and 11). This predictive capability can be used for disease treatment and/or prevention.

Described adaptive and personalized multivariable system identification approach exhibits following characteristics:

1. It enables a continuously "learning" system through adaptation to changes. The coefficient estimate vector $\hat{\boldsymbol{w}}$ provides customized correlation values for a single patient. The estimation error can be large initially but it decreases as the systems learns more ($\hat{\boldsymbol{w}}$ converges over time).
2. y and $\boldsymbol{u}$ are measured over time and $\hat{\boldsymbol{w}}$ adapts to their fluctuations. In some cases, $\hat{\boldsymbol{w}}$ "tracks" them in real time and constantly attempts to converge to new values. Since $\hat{\boldsymbol{w}}$ is no longer fixed, the least squares algorithm (Eq. 5) for conventional multiple linear regression cannot be used. Unlike conventional multiple linear regression which requires batch analysis, this adaptive approach uses only immediate previous y and $\boldsymbol{u}$ for estimation reducing memory usage.
3. Since $\hat{\boldsymbol{w}}$ can change over time, this variability can capture the non-linear dynamics of the relationships between dependent ($y[i]$) and independent ($y[i-1], \ldots, y[i-M], x_1[i-1], \ldots, x_1[i-N_1], \ldots, x_n[i-1], \ldots, x_n[i-N_n]$) variables which is assumed to be linear when the coefficients are fixed.
4. Unlike multiple linear regression that demands several statistical assumptions described earlier (e.g., normality of the error distribution), adaptive estimation techniques can deal with statistical variations.
5. Since this is a personalized approach, the solution from population-based studies to control confounding, such as randomization, restriction, and matching, cannot be used. The only way of removing the effect of any confounder is finding out and adding it to the study as an independent variable through medical knowledge-based and/or brute-force search.
6. As mentioned earlier, time delay plays a critical role in physiological dynamics. For example, $x_1[i-1]$ and $x_1[i-60]$ need to be treated as two separate potential confounding variables although they both belong to

x_1. In case $x_1$represents heart rate, the heart rate measured 1 minute ago ($x_1[i-1]$) can be correlated with current glucose level ($y[i]$) but the heart rate measured 60 minutes ago ($x_1[i-60]$) may not be so.

Distributed and adaptive multivariable system identification for artificial pancreas

The continuous glucose monitoring sensor data can be unreliable due to sensor deterioration/failure, unexpected communication interruption, significant environmental disturbance, etc., which can negatively influence or even disable the system's learning process. A distributed approach that enables "collective learning" can be considered for improved performance and greater resilience to failure. Privacy and secrecy are other reasons that promote distributed implementation. Patients may not be comfortable sharing their medical data with a remote fusion center that is vulnerable to cyber-attacks and can be a critical point of failure. In addition, as the number of patients increases, the repeated exchange of data back and forth between each patient data source and the fusion center can be costly and time-consuming.

The analysis and design of a distributed multi-agent network for collective optimization, adaptation, and learning from real-time streaming data have been studied in diverse fields of science and engineering, including swarm robotics, social and economic sciences, and biological sciences. Nature provides numerous examples of distributed networks, such as fish schooling, bee swarming, and bird flight formation, exhibiting sophisticated and intelligent behavior that emerges from interactions among agents and with the environment. I previously demonstrated this collective intelligence using a population of effector helper T cell, which is a type of immune cells that senses biological attacks such as bacterial infection at an early stage. My simulation and analysis suggested that these cells communicate with each other and make "collective decisions" more reliable than individual decisions prone to error and failure, while individual cells exhibit adaptive behavior.

There are three families of distributed strategies reported previously: (a) incremental strategies, (b) consensus strategies, and (c) diffusion strategies. In this proposal, I will focus on diffusion strategies which outperforms the other two strategies in the context of adaptive networks. Adaptive networks consist of distributed agents with adaptation and learning abilities. These agents are linked together through a specific topology and they interact with each other locally to solve inference and optimization problems in real time. Each agent shares information with its close neighbors to improve its learning process. The continuous sharing and "diffusion" of information through "overlapped" local interactions enable the agents to respond continuously to drifts in the data and to changes in the network topology. These networks are also robust to node (agent) and edge (connection between two nodes) failures and easily scalable. Diffusion strategies have been used to implement adaptive sensor networks in communications engineering and to model biological adaptation. Biological examples include bird flight formation, honeybee swarming behavior, and bacterial motility. To the best of my knowledge, diffusion strategies have not yet been applied to the artificial pancreas.

A network with 9 nodes is illustrated in **Figure 3A**. Each node representing a single patient can store and process data. Nodes connected by edges can share information with each other. Note these distributed nodes can exist in diverse places, including hospital electronic health record systems, cloud computing environments, and mobile/wearable devices. The multi-dimensional distance between two nodes is determined by the differences and similarities between two patients. For example, the distance between two patients with similar age, weight, height, past medical history, family history, social history, etc. will be very short. Since shorter distance is associated with less confounding, diffusion strategies should allow interactions between immediate or close neighbors only for better performance. Personal information can diffuse through overlapped localized interactions and may eventually affect the whole network in a dynamic fashion. For example, although red and blue sharing groups in **Figure 3A** are far apart without direct communication, nodes 4 and 6 mediate information flows so that these two distant groups can share information. However, this diffusion mechanism does not necessarily lead to one global consensus due to constantly fluctuating local dynamics.

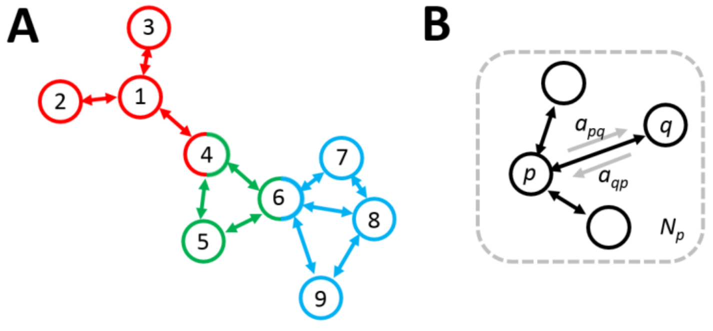


Figure 3. (A) 9-node network. (B) *p* node neighborhood (N_p).

The distance between any two nodes are not fixed and can dynamically change over time as the patient data evolve. This may result in changes in the network topology. The network topology can also be adaptively modified through re-wiring to achieve better performance. For example, a neighbor node that constantly provides unacceptable information can be disconnected and replaced by a more reliable node. **Figure 3B** illustrates the neighborhood of node *p* (denoted by N_p) that consists of all nodes connected to *p* by edges. It is shown in the figure that a pair of nonnegative weights a_{pq} and a_{qp} are assigned to the edge connecting *p* and its close neighbor *q*. From this figure, a diffusion strategy using LMS can be expressed as:

$$\begin{cases} \boldsymbol{\psi}_p[i+1] = \widehat{\boldsymbol{w}}_p[i] + \mu \boldsymbol{u}_p[i] e[i] \\ \widehat{\boldsymbol{w}}_p[i+1] = \sum_{q \in N_p} a_{qp} \boldsymbol{\psi}_q[i] \quad \left(\sum_{q \in N_p} a_{qp} = 1 \right) \end{cases} \qquad [13]$$

The first equation of Eq. 13 involves the adaptation process of node *p* (similar to Eq. 11) and assigns the result to a temporary variable $\boldsymbol{\psi}_p$. The second equation combines these temporary values from the neighbors and the node to produce the final $\widehat{\boldsymbol{w}}_p$. The weight a_{qp} can be determined by the distance between the nodes while the sum of all weights should be equal to one. Note the temporary variable $\boldsymbol{\psi}_p$ is used to link these two equations. This distributed and adaptive multivariable system identification approach can be coupled with model predictive control described earlier to enable personalized artificial pancreas.

One of my current projects aims to develop an experimental platform that uses adaptive system identification and feedback control to modulate the protein dynamics of "distributed" living cells. The goal of this project is to study the adaptive behavior of cell networks. In collaboration with Prof. Yongku Cho (University of Connecticut Chemical and Biomolecular Engineering), I'm harnessing optogenetics that uses light to control the synthesis and degradation of genetically-engineered proteins. Our platform uses Digital Micromirror Device (DMD), an addressable array of million microscopic mirrors, to give tailored light (color and intensity) to individual cells in parallel. Interestingly, the mathematical approach being studied for this experimental platform is similar to the one presented in this proposal and I believe there will be strong synergistic effects between these two projects.

Microservices-enabled artificial pancreas

One of the goals of the proposed distributed and adaptive approach is to find out the key independent variables, from a pool of virtually infinite number of choices, significantly correlated with glucose concentration. These correlations are not fixed and can dynamically change over time as relevant biologcial, psychological, social, and environmental factors change. Therefore, the data model $\boldsymbol{u}[i]$ (Eq. 8) introduced previously should be easily changeable to reflect these variations. In addition, we might want to use different adaptive system identification algorithms (e.g., LMS, RLS, Kalman filter, etc.) depending on given requirments and constraints that may also change over time. This need for flexibility leads us to the use of microservices-enabled apporaches.

The key concept of microservices is to decompose applications into independent, loosely-coupled small building block components or services that are easily changeable (or replaceable) and independently scalable without impacting other services. A conventional object-oriented application is "monolithic" and it is typically composed of tightly-coupled functional layers, such as user interface, logic, and database (**Figure 4A**). A monolithic application can be scaled by cloning it on multiple computers/virtual machines/containers (**Figure 4B**). On the other hand, a microservices application separates functionality into distinct smaller services or microservices (**Figure 4C**). This microservices approach scales out by deploying each service independently, creating instances of these services across computers/virtual machines/containers (**Figure 4D**).

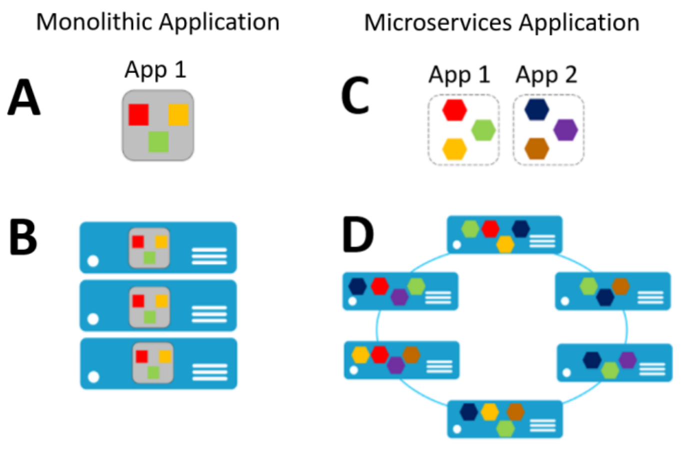


Figure 4. Monolithic application vs. microservices application.

Designing with a microservices approach is not a panacea for all projects. Since microservices communicate with each other through the network, communication latency and overhead due to the complex interaction paths can be problematic. Furthermore, simultaneous communication and processing of multiple microservices can bring the issues from distributed and parallel computing, including race condition and communication deadlock. I previously used MPI (Message Passing Interface) to simulate the parallel and distributed behavior of adaptive multicellular gene networks. For the simulaiton, 3,500 compute nodes from the TACC (Texas Advanced Computing Center) Stampede supercomputing cluster were used (each node represented a single cell). These supercomputing clusters were "closed" systems for increased performance and security, which prevented any cluster application from communicating with external streaming data sources across the internet. In this context, I explored cloud computing and received the Microsoft Azure Research Award for two consecutive years (2014 and 2015) along the path of the journey. However, the unreliability due to unexpected virtual machine failures and the resulting loss of states was a significant concern for communication-intensive distributed applications. Thankfully, this problem was recenlty resolved by microservices PaaS (Platform-as-a-Service), which orchestrates microservices (and the containers holding them) to achieve reliable and stateful microserivces. Microservices PaaS includes open-source Kubernetes and Service Fabric from Microsoft.

This project will use Service Fabric which makes it easy to package, deploy, and manage reliable and scalable microservices. Microsoft used Service Fabric internally for more than 7 years to support their cloud applications, including Skype for Business, Azure SQL Databases, Cortana, etc. Using Service Fabric, developers can avoid complex fundamental cyberinfrastructure issues (microservice discovery, failover, replication, state management, health monitoring, load balancing, etc.) and focus on the application logic, which can dramatically increase productivity. Vendor lock-in is becoming less of an issue as vendors are increasingly making their tools programming language independent, operating system independent, and open source. In addition, the same code can be deployed on private, hosted, or public clouds (Amazon Web Service, Microsoft Azure, etc.) as the code is wrapped within a container providing the same code environment. Service Fabric also supports reliable and stateful microservices-enabled actors. The Actor pattern (also known as the Actor architecture or the Actor model) provides a powerful tool to abstract complex distributed systems in which a large number of independent agents collaborate. Actors are isolated, independent units of logic and state (i.e., every actor has its own storage) within a single-threaded execution model. Concurrency is achieved by having many actors executing simultaneously and communicating with each other asynchronously. Service Fabric does not automatically solve inherent complexities such as deadlock but provides tools to deal with them, thus increasing productivity.

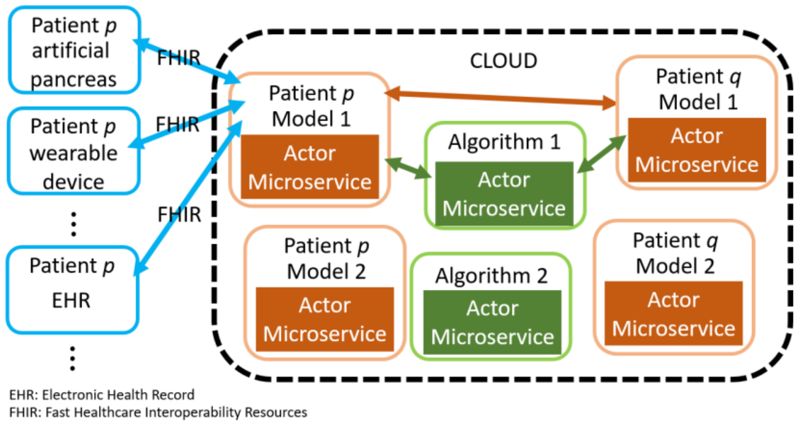


Figure 5. A microservice-enabled actor model for distributed and adaptive personalized artificial pancreas.

Figure 5 illustrates a microservice-enabled actor model for distributed and adaptive personalized artificial pancreas. Each data model (Eq. 8) for a specific patient (*p* in the figure) is represented by a stateful actor microservice instance which communicates with external data sources using the Fast Healthcare Interoperability Resources (FHIR) standard. FHIR was developed by the healthcare IT standards body known as Health Level 7 International (HL7) and provides a comprehensive standardized framework for the exchange, integration, sharing, and retrieval of electronic health information. Different adaptive system identification and control algorithms are also represented by distinct actor microservice types. Note Patient *p* actor microservice and Patient *q* actor microservice are two instances of the same microservice type (Model 1). Microservice type and instance can be considered as class and object instance in object-oriented programming. Although not shown in the figure, the actor microservices communicate with external data sources via Azure IoT (Internet of Things) Hub introduced later.

Preliminary work (distributed and adaptive multivariable system identification)

Collaboration with neighbors may save a patient when the learning capability is unexpectedly lost. For example, assume that there are four actor microservice instances (each actor microservice instance representing a type-1 diabetic patient) and the microservice instance for patient *p* suddenly loses its learning capability (red arrow in **Figure 6**) while maintaining communication with its neighbors. The model for each patient can be expressed as:

$$y[i] = \beta_0[i] + \alpha_1[i]y[i-1] + \alpha_2[i]y[i-2] + v[i] \qquad [14]$$

where y is the glucose level measured every five minutes. Eq. 14 is basically stating that the current glucose level $y[i]$ depends on the level 5 minutes ago ($y[i-1]$) and the level 10 minutes ago ($y[i-2]$). **Figure 6** shows that the microservice instance for patient *p* can still "learn" and make relatively sound predictions ($\hat{y}[i+1]$) using the coefficients estimates from its neighbors (yellow line in the figure) when in cooperative mode. Note the prediction errors are large in the beginning but decrease as the microservice learns (dark blue line in the figure). In contrast, non-cooperative mode results in poor prediction (light blue line in the figure). A Service Fabric cluster providing 5 virtual nodes installed on a laptop (Windows 7 64-bit, processor speed: 2.50 GHz, RAM: 8 GB) was used for this simulated experiment. Service Fabric automatically distributes and manages the four actor microservice instances and their duplicates across the five nodes. This preliminary work is based on the online datasets from a study titled "A Randomized Clinical Trial to Assess the Efficacy of Real-Time Continuous Glucose Monitoring in the Management of Type 1 Diabetes".

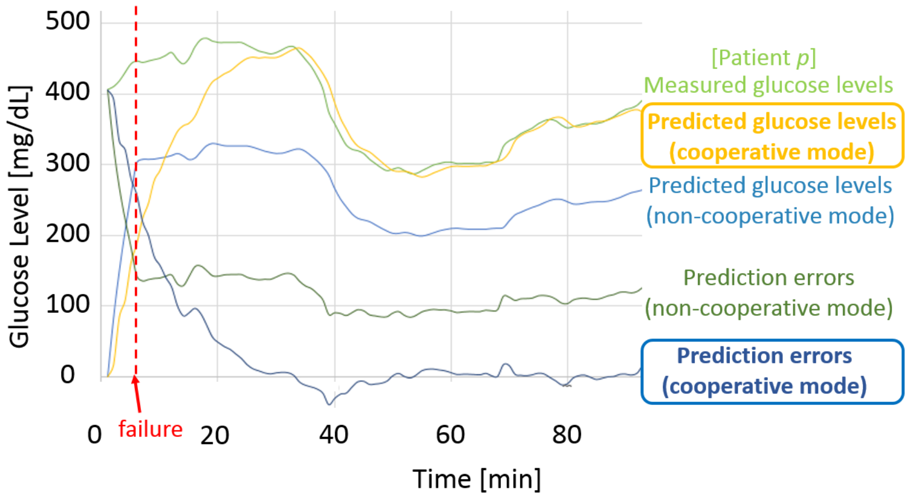


Figure 6. Cooperative vs. non-cooperative mode.

Edge computing for artificial pancreas

For the construction of the artificial pancreas, this project will use the system architecture that the Open Artificial Pancreas Project (OpenAPS) has developed. The architecture mainly consists of three components: 1) continuous glucose sensor, 2) single-board computer such as Raspberry Pi, and 3) insulin pump (**Figure 7**). Windows IoT (Internet of Things) Core Operating System (OS) will be installed on the single-board computer to enable edge computing. Edge computing is an approach of delivering cloud capabilities to the local (edge) devices to reduce the communication burden and to cope with the loss of network connectivity. Several vendors are currently providing edge computing service, including Microsoft Azure IoT Edge and Amazon Web Service Greengrass. This project will use Azure IoT Edge that enables cloud code and services to be securely delivered to local artificial pancreas systems, thus distributing intelligence from the cloud to the edge. Local artificial pancreas systems supported by Azure IoT Edge can communicate with a Service Fabric application described earlier through Azure IoT Hub, a fully managed service that enables reliable and secure bidirectional communications between millions of edge devices and the cloud (**Figure 7**). Azure IoT Hub has a device twin for each edge device connected to the hub. This means there will be a device twin in the cloud that contains the same model and algorithm for every artificial pancreas. Whenever this twin is updated with an optimal model and/or algorithm by distributed and adaptive learning explained before, the change will be delivered to the corresponding artificial pancreas so that it can perform with the latest update even when network connectivity is lost. For increased productivity, a unified development platform (Microsoft Visual Studio) and programming language (C#) will be used for the development of both cloud and edge applications.

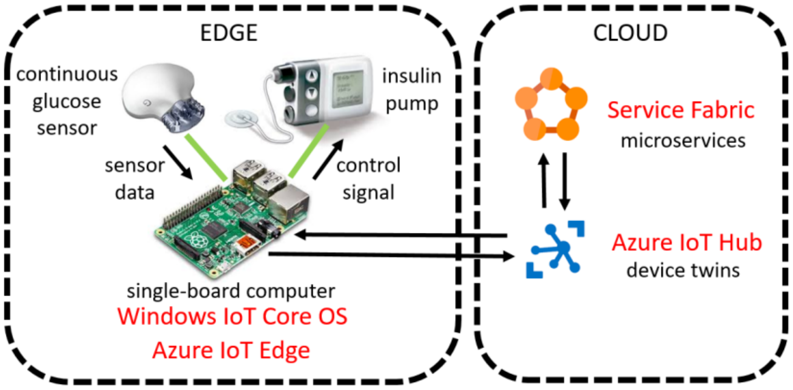


Figure 7. Edge computing for artificial pancreas.

***In silico* (computer-based) preclinical trial**

Recently, the lengthy and costly preclinical animal studies for the artificial pancreas have often been replaced by *in silico* (computer-based) simulations. This project will use the University of Virginia-Padova type 1 diabetes mellitus simulator (T1DMS) for simulated experiments. T1DMS is a computer model of the human metabolic system based on the glucose-insulin dynamics in human subjects. The T1DMS technology provides realistic computer simulation of clinical trials using a population of 300 subjects with parameters derived from triple tracer metabolic studies that reflect the human metabolism found with type 1 diabetes. It has been validated against actual clinical data and is accepted by the FDA as a substitute for pre-clinical animal trials in the testing of certain control strategies, including feedback control algorithms, for type 1 diabetes. **Figure 8** schematically illustrates the preclinical study design. T1DMS basically provides continuous glucose sensor data and responds to control signals from the single board computer that communicates with the cloud.

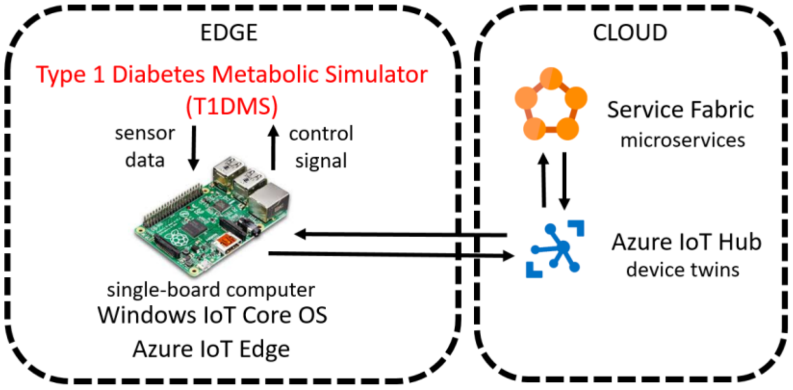


Figure 8. *In silico* preclinical trial.

Research questions

- Cooperative vs. non-cooperative mode of operation: Will individual agents always benefit from cooperation? Can we develop optimized combination policies that enable all gents in a network to improve their performance through cooperation?
- Performance: What are the factors that can affect performance across the network?
- Distance/weighting: How can we define the distance between two nodes? What are other factors which can affect weighting?
- Topology: How can immediate neighbors be determined? Can we develop criteria for determining which and how many neighbors to interact? Will more neighbors always bring better performance? How can adaptive topology be implemented?
- Idle agents and link failures: How can idle agents and link failures be managed?
- Asynchronous communication: Asynchronous communication is preferred within an uncertain environment like the internet. However, asynchronous communication can cause irregular communication delay. Will this irregular delay degrade performance? What will be the tolerable range of delay?
- Sampling: The sampling interval for measuring each variable can be different and irregular (the time elapsed between two consecutive measurements may vary). How can this issue be addressed?
- Privacy and security: Although Microsoft Azure supports the HIPAA (the Health Insurance Portability and Accountability Act) and HITECH (the Health Information technology and Economic and Clinical Health) Act compliance, are there any privacy and security concerns?
- Data types: In medical research, there are mainly three types of data: 1) quantitative (continuous or discrete), 2) categorical (binary, nominal or ordinal), and 3) time-to-event data types. Only continuous quantitative data type is discussed here. How can the remaining data types be incorporated into the proposed framework?
- Can existing diabetes management platforms, such as Tidepool (tidepool.org) and Glooko (glooko.com) which supports real-time glucose sensor and insulin pump data sharing, be integrated with the proposed platform via REST (Representational State Transfer) API (Application Programming Interface)?
- Can REST APIs be developed so that a developer/researcher can readily build customized distributed and adaptive personalized artificial pancreas?

Evaluation plan

Performance measures include estimation/prediction error, convergence rate, stability, percent time in euglycemic range, and frequency of hypoglycemic events. All tools and documentation will be made available on GitHub under the MIT license. The size and activity of this community will be used to measure the success.

Collaborators

Prof. Ali Cinar (Illinois Institute of Technology)
Prof. Adi Saleh (UCSF) and Prof. Jenise Wong (UCSF): Founders of TidePool (a cloud-based, open-source, not-for-profit diabetes data management company)
Haishi Bai (Microsoft): Principal software engineer and technical advisor to Azure CTO

Related project

In 2017, I received a 2-year grant titled "Distributed and Adaptive Personalized Medicine" from NSF (the National Science Foundation) SCH (the Smart and Connected Health Program):
https://www.nsf.gov/awardsearch/showAward?AWD_ID=1723483&HistoricalAwards=false