

Real-Time Control of Soil Moisture for Efficient Irrigation

by
Samantha Harper

B.S. Environmental Engineering Science

Massachusetts Institute of Technology, 2016

Submitted to the Department of Civil and Environmental Engineering
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Environmental Engineering

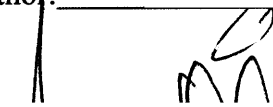
at the
Massachusetts Institute of Technology

June 2017

© 2017 Massachusetts Institute of Technology. All rights reserved.

Signature redacted

Signature of Author: _____



Samantha Harper
Department of Civil and Environmental Engineering
May 12, 2017

Signature redacted

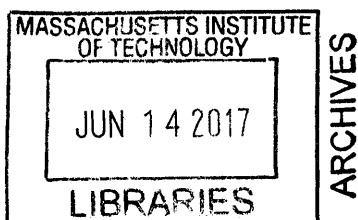
Certified by: _____

Dennis McLaughlin
H.M. King Bhumibol Professor of Water Resource Management
Department of Civil and Environmental Engineering
Thesis Supervisor

Signature redacted

Accepted by: _____

Jesse Kroll
Professor of Civil and Environmental Engineering
Chair, Graduate Program Committee



Real-Time Control of Soil Moisture for Efficient Irrigation

by
Samantha Harper

Submitted to the Department of Civil and Environmental Engineering
on May 12, 2017 in Partial Fulfillment of the Requirements for the
Degree of Master of Engineering in Environmental Engineering

Abstract

In the field of precision irrigation control, two classes of controllers have emerged – classical controllers and model based controllers. The most widely-used real-time closed-loop controller is a bang-bang controller that applies water at a predetermined rate, duration, and minimum soil moisture. Due to the ease of installation of soil moisture sensors, this technology has been installed around the world. There have been few studies on altering the controller used with this existing infrastructure. This thesis articulates a model for using a real-time proportional-integral-derivative (PID) controller to minimize water use using HYDRUS-1D, a software package for simulating the one-dimensional movement of water, heat, and solutes in porous media, to simulate soil moisture. In a direct comparison between the two controllers, the PID controller uses less water. However, small violations of the target soil moisture and optimization of the PID parameters present the current barrier to implementation of this technology. Maintaining soil moisture at or above minimal depletion is critical to support crop health throughout a growing season. PID controllers offer a mid-point between the simplistic bang-bang controllers and the model based controllers that require large datasets, wireless network infrastructure, and robust computing systems. With proper calibration, PID controllers can be implemented in the field with the same sensors that are widely used with bang-bang controllers resulting in a reduction of water use in regions where water is scarce.

Thesis Supervisor: Dennis McLaughlin
Title: H.M. King Bhumibol Professor of Water Resource Management
Department of Civil and Environmental Engineering

Acknowledgements

Thank you to my teachers, friends, and family who have supported me through this journey.

To Dennis, thank you for your encouragement, guidance, constructive criticism, and willingness to talk through anything.

To “the group,” thank you for your support and friendship throughout this exciting graduate school adventure.

To my family, thank you for listening to me talk about this for a year and for agreeing to read and edit my paper ‘one last time’ before submission.

To my friends, thanks for being there when I needed a break, or a hug, or some chocolate.

To my mentors and professors, thank you for pushing me to be better than I thought I could be and showing me what successful, compassionate leaders look like.

Contents

Abstract	3
Acknowledgements.....	5
Introduction.....	9
Motivation.....	9
Research Question and Objectives.....	9
Literature Review.....	10
State of Irrigation Control Systems.....	10
HYDRUS-1D for Irrigation Optimization.....	14
Theoretical Framework	14
HYDRUS-1D.....	15
Bang-Bang Controller.....	17
PID Controller.....	18
Methods.....	18
Experiment.....	19
Historical Data Collection.....	19
HYDRUS-1D Set-up	20
Code Structure	21
Finding nominal irrigation – Bang-Bang Controller Optimization	21
Finding Optimal PID Parameters.....	23
Results.....	25
Discussion	29
Conclusion	30
Bibliography	31
Appendix 1: Data Collection	35
FAO Crop Factors.....	35
High Plains Regional Climate Center Stations	35
Acres of Irrigated Land in 2012.....	36
Appendix 2: Code from Python.....	37
Appendix 2: Code from MATLAB.....	47

Introduction

Motivation

Globally, agriculture uses 70% of all water withdrawn from aquifers, streams, and lakes (FAO, 2011). Water withdrawals for industrial, livestock, and domestic use are projected to increase 50% by 2025 which will limit irrigation withdrawal (Rosegrant et al. 2002). At the same time food production capability must increase 50-100% to fulfill the demands of the growing population (Alexandratos and Bruinsma 2012). With a changing climate, population growth, and increasing demand from many industries, the availability and distribution of freshwater resources becomes more unpredictable every year. This situation demands that water resources be used efficiently and intelligently to reduce water losses and increase water productivity (FAO, 2012). Over the last 40 years, the study of precision agriculture has developed tools and technology which have expanded the data available to increase the profit and sustainability of global agriculture. Field and remote sensing provide decision-makers with unprecedented amounts of information. Enhanced machinery allows for specificity in allocation of fertilizer and irrigation. Information systems and management tools work in collaboration with, improve, or replace historical intuition-based decision-making to optimize the distribution of water and fertilizer resources. Improved irrigation scheduling and increased irrigation efficiency are priorities for the sustainability of our global food system. To maintain the quality of the environment and improve the sustainability of global food supply, the agricultural industry must increase production using fewer resources. (Gebbers and Adamchuk 2010; McBratney et al. 2005; Pereira et al. 2002; Stafford 2000).

In climates with irregular, uncertain rainfall, real-time control plays a large role in the future of precision agriculture and water resource management. Precision irrigation aims to apply water when and where it is necessary to support plant growth while decreasing excess runoff and minimizing water use. Control theory is an interdisciplinary branch of mathematics and engineering developed to control dynamical systems. It aids in complex decision-making processes and deals with uncertainty in a quantified and precise manner. Optimization of irrigation controllers can provide efficient irrigation to maintain crop health and yield. This thesis analyzes two control methods, bang-bang and proportional-integral-differential, for scheduling irrigation. These methods are evaluated with a one-dimensional simulation of water movement in porous media (HYDRUS-1D) as a test field site.

Research Question and Objectives

This thesis develops a process for tuning a real-time controller based on historical weather data to minimize water used for irrigation. A proportional-integral-derivative (PID) soil moisture controller is compared to a traditional soil moisture control method. Meteorological data from

southeastern Nebraska are combined with a one-dimensional soil column simulation that acts as a surrogate field site.

Literature Review

Irrigation optimization has been a goal of precision agriculture since its inception. Automated real-time irrigation control systems lower labor costs, plant stress levels, and water use (Evetts and Howell 2000). The literature reveals a gap in the research between classical controllers, such as bang-bang and PID, and complex controllers, such as model predictive control and artificial intelligence, for irrigation. This thesis analyzes the classical controllers, bang-bang and PID, for their effectiveness in real-time irrigation scheduling using HYDRUS-1D. There have been many studies that show that HYDRUS-1D provides an accurate model of soil moisture and matric head distribution, when compared to field measurements, and can therefore be used as a surrogate field site for irrigation scheduling.

State of Irrigation Control Systems

Irrigation scheduling can be based on sensor measurement of either soil water content or plant stress. The major parameters that determine irrigation requirements are crop type, stage of growth, soil characteristics, atmospheric conditions, and water balance. Irrigation control based on plant stress measurement does not indicate how much water to apply and requires extensive calibration to determine control thresholds (Jones 2004). The most appropriate measurements for minimizing water use are soil moisture and matric head distribution, which directly relate to the amount of irrigation applied to the crop and the biomass yield (Shani et al. 2004). For these reasons, this thesis focuses on soil water measurements.

Soil water sensors can measure two parameters: matric head and soil water content. Matric head is measured by tensiometers and psychrometers. Soil water content is measured by capacitance probes, time-domain reflectometry, neutron probes, gravimetric sampling, gamma ray attenuation, gypsum block electrochemical cells, pressure plates, and ground penetrating radar (Dobriyal et al. 2012; Zazueta and Xin 1994). Capacitance probes, time-domain reflectometers, neutron probes, and dielectric probes are easy to deploy, can be quite precise, and are widely available with threshold (bang-bang) irrigation control in commercial systems (Greenwood et al. 2010). It should be noted that soil heterogeneity requires many sensors to accurately depict the entire field (Jones 2004). Two and three dimensional analyses have been developed to understand the placement of soil moisture sensors to overcome the barrier of heterogeneity (Dabach et al. 2013; Mailhol et al. 2011; Seidel 2015; Soulis et al. 2015).

There are two basic approaches to control irrigation scheduling: open-loop control and closed-loop control (Adeyemi et al. 2017; Dabach et al. 2013; McCarthy et al. 2013; Romero et al. 2012; Zazueta and Xin 1994). Open-loop control allocates irrigation water without receiving any feedback from the controlled system. In irrigation, this type of controller requires a specified amount of water, irrigation period, and number of cycles over the course of a season. Typically, these are based on historical data for rainfall, evapotranspiration, and yield. Experts claim that applying large amounts of water at periodic intervals is better for the soil chemistry as it “washes the soil free of chemicals and creates a better balanced soil” (Bahat et al. 2000). These are the cheapest and most readily-available irrigation management systems but they do not provide the optimal solution for irrigation because they do not adapt to changes in the system (Sarwar et al. 2001). Closed-loop control allocates water at each time step based on feedback from the system. These controllers can vary the flow rate, irrigation period, and time between applications. The closed-loop controllers work through three modalities: to achieve and maintain optimal water content throughout the growth period of the crop; to irrigate when plants are stressed; and – coupled with crop production models – to optimize yield, profit, or water productivity. As of 2013, there were no published comparisons of performance between these closed loop modalities (McCarthy et al. 2013) and this literature review did not find any such comparison.

The most basic closed-loop controller is the bang-bang controller (also known as the on-off, triggered, or threshold control). Classical control systems do not rely on any particular model of the system; they respond to measured inputs (information that can be gathered by sensors) instead of a complex model. Depending on the sign of the error between the specified target and a measured input, the controller switches between a maximum and minimum output. This is the most widely-used closed-loop controller on the market. Most commercial controllers operate on this control method including Acclima, Watermark, Rain bird, and Water Watcher. They require specification — of the amount and duration of irrigation — and solely control when that irrigation is released. Bang-bang controllers which utilize a long cycle between irrigation releases create large deviations from the target while those with short cycles between releases cause excessive wear on the distribution system and increased water use (Romero et al. 2012). Thus, there is a trade-off between cycle length and deviation from the target. Many studies have shown that these controllers can reduce water use compared to open loop controllers. In Gainesville, FL, four soil moisture based sensors with bang-bang control skipped an average of 71% of irrigation cycles scheduled by an open loop controller (Cardenas-Lailhacar and Dukes 2010). Also in Florida, a dielectric soil water probe saved 61% and tensiometer paired with a bang-bang controller saved 79% of water when compared to an open loop fixed irrigation schedule based on historical evapotranspiration data (Muñoz-Carpena et al. 2008). Bang-bang control can also assist with reducing nutrient leaching. In addition to reducing water use by 7% to 62% from a fixed time irrigation schedule, a bang-bang controller reduced nitrogen leaching by 25% to 73% in Raleigh, NC (Zotarelli et al. 2011). In the field, there has been more work on optimizing the bang-bang controller (varying thresholds, correctly positioning sensors, and

determining the proper calibrations for each implementation) than on studying alternate control strategies. A controller that further reduces water use and maintains yield would be a promising addition to precision irrigation.

Classical control systems like proportional-integral-derivative (PID) control perform more complex calculations with the single input. The PID control signal is a weighted sum of the error between the target and the measured output from the system, the integral of recent errors, and the rate at which the errors have been changing. The PID controller has not been widely used in commercial controllers even though there are a number of commercially available variable-rate irrigation techniques. The greatest limitation to the adoption of variable rate technology is associated with developing optimal irrigation schedules (volume and timing of irrigation) and irrigation prescriptions (Evans et al. 2013; McCarthy et al. 2013). PID controllers could be adapted to not only follow a single threshold value, but an evolving threshold as the plant water needs change over the season.

This literature review revealed two papers that used PID controllers with precision irrigation. It should be noted that PID control has been simulated for application in open channel flow in irrigation channels based on their hydraulic properties (Aguilar et al. 2016; Álvarez et al. 2013; Bolea et al. 2014a; b; Lacasta et al. 2014; Lozano et al. 2010); this does not include models for crop water demand or soil moisture. Further research in peer-reviewed journals on the use of PID controllers in precision irrigation is scarce. Romero et al. 2012 compared a PID controller to a more a model predictive controller, which is more complex. This study found that the model predictive controller achieved the set point faster and remained closer to the target soil moisture because of its ability to adapt to the irrigation needs in advance with predictions of precipitation. At the Institute of Electrical and Electronics Engineers International Conference on Computer and Information Technology in 2014, Bi and Zheng presented a paper on the use of grey fuzzy PID control for the flow of water and fertilizer, establishing the appropriate controller model to determine the control parameters of the system. This has proven to achieve effective results in reducing water use and fertilizer runoff.

Model-based control strategies are not considered in this thesis; they are mentioned to give a larger overview of the literature surrounding irrigation control theory. These methods are not commercialized and require much more infrastructure (sensors and networking) to operate. They are a growing area of research because it can be hard to find a soil water content trajectory to maximize yield, water use efficiency, and profit without consideration of the entire soil-crop-atmosphere system (Romero et al. 2012). These strategies include fuzzy logic control, genetic algorithms, neural networks, model predictive control, linear quadratic control, and learning control. These model-based controllers require more information than just the input and output of the model to make decisions and are ideal for highly nonlinear systems. The literature suggests

that a soil-plant-atmosphere model is more useful in conjunction with these controllers than a “black box” crop model (McCarthy et al. 2013). One controller widely used in machine control is the fuzzy logic controller, which analyzes analog input in a range from 0 to 1 rather than assigning a binary value. This means that a fuzzy logic controller can assess values as more than just “true” or “false” and include a “maybe” or “partially true” assessment in the output (Romero et al. 2012). Examples of fuzzy logic controllers can be found in Bahat et al. 2000, Giusti and Marsili-Libelli 2015, and Xiang 2011.

Genetic algorithms can be used to construct numerical optimization techniques that “evolve” towards better solutions by taking the best of one set of solutions and generating new inputs to see if those are better than the current “fittest” solutions (Romero et al. 2012). A neural network is able to capture complex input/output relationships. This structure allows for flexible representation of non-linear functions where input and output can rely heavily on one another. This is a modelling framework that can be used in support of advanced model-based controllers like those found in Kolassa et al. 2013 and McClendon et al. 1996.

Model predictive control solves many open-loop optimal control problems based on a model of the system, and determines the optimal trajectory for the output. As new information becomes available the controller updates its forecast of the system response and derives a new optimal trajectory based on the updated forecast. Model predictive control works best in systems that can be highly regulated, like greenhouses (Adeyemi et al. 2017; McCarthy et al. 2013; Romero et al. 2012). Park et al. 2009 applied a receding horizon control scheme that demonstrated a viable strategy for minimizing negative environmental impacts and achieving water reuse for a sprinkler system in California. The VARIwise simulation framework is an example of model predictive control (McCarthy et al. 2014). The amount of data required for such a model limits the use of VARIwise and other software like it to high value crops (Adeyemi et al. 2017).

Linear quadratic control is limited by its requirement that the system response must be linearly related to the control variable. This requirement is satisfied only in an approximate sense in irrigation applications. Linear quadratic control can generate an optimal soil moisture curve soil moisture curve for the season rather than deciding when and where to irrigate (McCarthy et al. 2013). Examples of linear quadratic control can be found in Protopapas and Georgakakos 1990 and Shani et al. 2004. Iterative learning control is used in systems that operate repetitively with the same initial conditions and can adjust irrigation flow rate to control water and nutrients in the soil based on past irrigation-crop response measurements (Adeyemi et al. 2017). Learning control requires that a single variable represents overall plant status and may require more irrigation to achieve optimal results than model predictive control because it lacks prediction from a crop model (McCarthy et al. 2013).

Before these methods can be widely implemented in the field, they must be robust, optimal, and stable. This is not currently the case for any individual model-based control method mentioned in this section (McCarthy et al. 2013). This thesis compares the classical controllers — bang-bang and PID — for real-time irrigation application; to determine targets and test performance of these controllers, HYDRUS-1D acts as a surrogate field site.

HYDRUS-1D for Irrigation Optimization

HYDRUS-1D is a software package for simulating the one-dimensional movement of water, heat, and solutes in variably-saturated media (Šimůnek et al. 2009). There are many studies indicating that HYDRUS-1D supplies adequate information on soil water content and variation in matric head such that it can be used to optimize irrigation controls in a field. HYDRUS-1D combined with the EPIC crop model was used to find the amount of irrigation that should be applied to reduce nitrogen runoff in the Northern China Plain for winter wheat under sprinkler irrigation (Wang et al. 2015). Another study found that HYDRUS-1D can accurately simulate flow in multi-layer paddy soil where the plow pan plays a role in determining vertical pressure head (Tan et al. 2014). This means HYDRUS-1D can be used for crops that require continuously flooded irrigation like rice. Tafteh and Sepaskhah 2012 found that HYDRUS-1D accurately simulated deep percolation of water for both rapeseed and maize with good accuracy in semi-arid regions. It also accurately simulated capillary rise in Uzbekistan and was used to develop an irrigation schedule to minimize water logging and optimize water use in the region (Akhtar et al. 2013). Sammis et al. 2012 showed that HYDRUS-1D worked reasonably well to schedule irrigation for deep-rooted crops when using drip irrigation in the southeastern United States. These papers are representative of the breadth of validation that HYDRUS-1D has received as an accurate replacement for field measurements in irrigation control.

Theoretical Framework

There are three components that make up the theoretical framework for this thesis. The first is a one-dimensional simulation, HYDRUS-1D, that takes inputs of irrigation, rainfall, soil parameters, and evapotranspiration and resolves the movement of water in the column. This is used to help determine a target irrigation rate and to simulate soil moisture in order to test the two control methods considered.

The first control method is a bang-bang controller. It checks the real-time soil moisture output and applies irrigation at a specified constant rate and duration. The second control method, a proportional-integral-derivative (PID) controller, checks the real-time soil moisture output and applies irrigation at a varying rate and interval. The real-time soil moisture information required

by both controllers is obtained from HYDRUS-1D rather than a real field observation. In this respect, HYDRUS-1D is used to simulate a field test.

Together, the HYDRUS-1D simulator and the two controllers provide insight into the optimization of irrigation for minimizing water use.

HYDRUS-1D

The HYDRUS program numerically solves the Richards equation (Equation 1) for variably saturated water flow and advection-dispersion type equations for heat and solute transport in unsaturated, partially saturated, or fully saturated porous media. For the initial set-up, HYDRUS-1D requires inputting depth of soil and soil properties, length of simulation, boundary conditions for the top and bottom layers, and daily evaporation, transpiration, and rainfall; and choosing a root water uptake model ($S(h, h_{\phi}, x) = \alpha(h, h_{\phi}, x)b(x)T_p$ Equation 2) and a hydraulic model (Equation 4) The soil properties – depth of profile, soil materials, and van Genuchten-Mualem parameters – are defined by the user based on empirical measurements. The simulation length depends on the growth of the crop. The boundary conditions can be defined as prescribed head and flux boundaries, boundaries controlled by atmospheric conditions, or free drainage boundary conditions. The daily evapotranspiration and rainfall are measured meteorological data. The following describes the equations that define the system used as a surrogate field site for this simulation.

The Richard's equation describes the relationship between changing soil moisture and the soil parameters defined by the user – hydraulic conductivity, K , and matric head, h .

$$\frac{\partial \theta}{\partial t} = \frac{\partial \theta}{\partial z} \left[K(\theta) \left(\frac{\partial h}{\partial z} + 1 \right) \right] - S \quad \text{Equation 1: Richard's Equation for Variably Saturated Water Flow}$$

Where K is the hydraulic conductivity [cm/day], h is the matric head induced by capillary action, z is the elevation head above the vertical datum, θ is the volumetric water content [·], t is time [days], and S is the sink term to account for water uptake by plant roots (Richards 1931).

The sink term, S , of the crop water uptake in Equation 1 is specified in terms of a stress factor defined by Feddes et al. 1978 and modified by van Genuchten 1980 (Equation 2).

$$S(h, h_{\phi}, x) = \alpha(h, h_{\phi}, x)b(x)T_p \quad \text{Equation 2: Feddes Crop Water Uptake}$$

Where

The soil properties required are depth of profile, number of soil materials, and van Genuchten-Mualem parameters. The matric head and volumetric soil moisture are related by the van

$= \theta_r + \frac{\theta_s - \theta_r}{[1 + \alpha|\psi|]^{1-1/n}}$ Equation 3). This equation allows the user to specify matric head to adjust soil moisture initial conditions in the HYDRUS-1D user interface.

$$\theta(\psi) = \theta_r + \frac{\theta_s - \theta_r}{[1 + \alpha|\psi|]^{1-1/n}} \quad \text{Equation 3: van Genuchten Water Retention Equation}$$

Where $\theta(\psi)$ is the water retention curve [·], $|\psi|$ is the suction pressure, θ_s is the saturated water content [·], θ_r is the residual water content [·], α is related to the inverse of air entry suction, and n is a measure of pore-size distribution (van Genuchten 1980).

In 1976, Maulem modified to the van Genuchten equation to account for pore size distribution in the hydraulic conductivity calculation. HYDRUS-1D uses the van Genuchten-Mualem equation (Equation 4) to simulate soil moisture in the column. It adds a new parameter, l , to account for the tortuosity in the conductivity function. By introducing the parameter l , one has the opportunity to influence the shape of $K(h)$ and add one more degree of freedom during the optimization, which is useful for well-defined experimental data, such as multistep outflow and evaporation methods. The Mualem equation defines hydraulic conductivity in the soil at saturated and unsaturated conditions in soils with varying pore sizes.

$$K = \begin{cases} K_s S_e^l \left\{ \frac{1}{2} \operatorname{erfc} \left[\frac{\ln \left(\frac{h}{\alpha} \right) + \frac{n}{\sqrt{2}}}{\sqrt{2}n} \right] \right\}^2 & h < 0 \\ K_s & h \geq 0 \end{cases} \quad \begin{array}{l} \text{Equation 4:} \\ \text{van Genuchten-Mualem Equation for} \\ \text{Predicting Hydraulic Conductivity in} \\ \text{Unsaturated Porous Media} \end{array}$$

Where K is hydraulic conductivity [cm/day], K_s is saturated hydraulic conductivity [cm/day], S_e is effective saturation [·], h is capillary height [cm], α (related to air entry suction) [1/cm] and n (a measure of pore size) [·] are both empirically derived coefficients (Mualem 1976; Šimůnek et al. 2009).

The outputs of the simulation are numerous, but the real-time controllers only use the values of volumetric water content ($\theta_m(t)$) returned at the end of each day that the simulation is run to determine the irrigation input ($u(t)$) for the following day this can be seen in Figure 1.

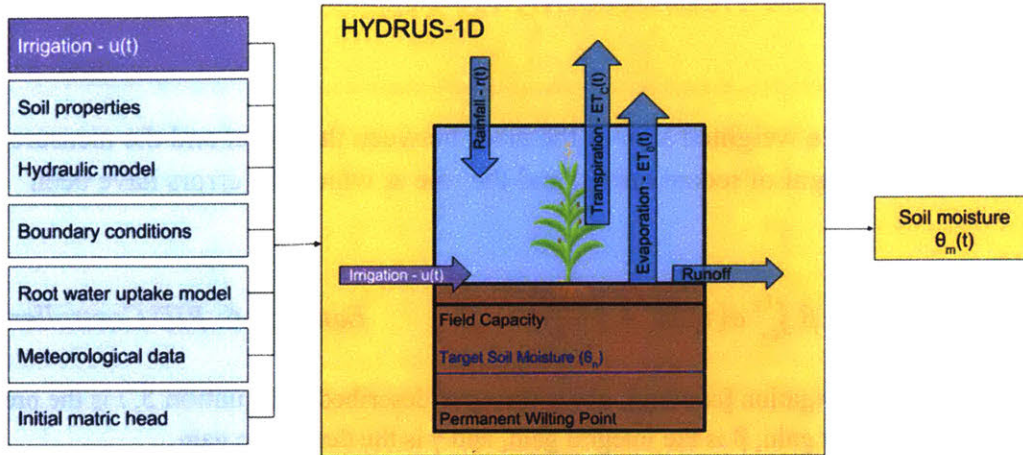


Figure 1: Relevant Inputs and Output of HYDRUS-1D Model

Bang-Bang Controller

The most basic closed-loop controller is the bang-bang controller. Depending on the sign of the error between the most recent measurement and a specified target, the controller switches between a maximum (some water) and minimum (no water) output.

$$\theta_m(t) - \theta_n = e(t)$$

Equation 5: Bang-Bang Controller

if $e(t) < 0$ Add water; $u(t) > 0$

if $e(t) \geq 0$ Do not add water; $u(t) = 0$

Where $\theta_m(t)$ is the measured volumetric water content at time t [·], θ_n is the target volumetric water content [·], $e(t)$ is the error – the deviation of the measured soil moisture from the target [·], and $u(t)$ is the irrigation applied [cm/day].

Figure 2 is a flow chart of the decision-making process of the bang-bang controller.

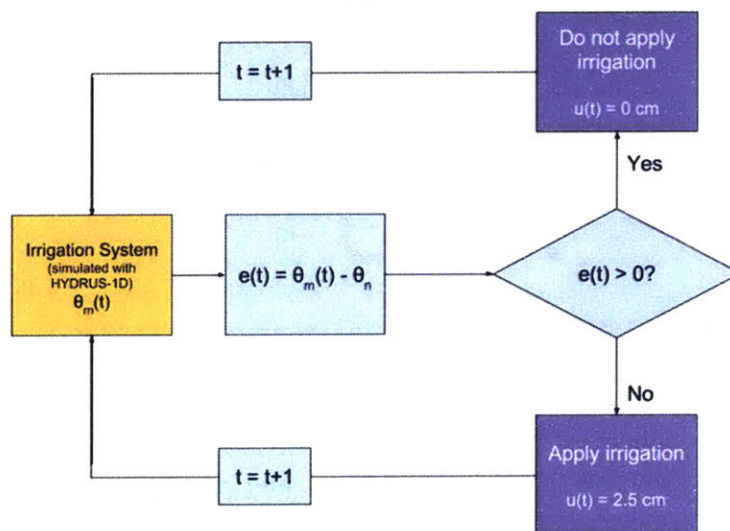


Figure 2: Bang-Bang Controller Decision Chart

PID Controller

The PID control signal is a weighted sum of the error between the target and the measured output from the system, the integral of recent errors, and the rate at which the errors have been changing.

$$u(t) = \alpha e(t) + \beta \int_{t_1}^{t_2} e(\tau) d\tau + \gamma \frac{de(t)}{dt} \quad \text{Equation 6: PID Controller}$$

Where $u(t)$ is the applied irrigation [cm/day], $e(t)$ is the error described in Equation 5, t is the present time step, α is the controller gain, β is the integral gain, and γ is the derivative gain.

Increasing α will increase sensitivity, making the response more oscillatory and the system less stable. Reducing β will increase the amplitude of oscillations, lengthen settling time, make the response more sluggish, and make the system less stable. Reducing γ will reduce settling time, speed up response, and amplify noise (Romero et al. 2012). These parameters can be adjusted to maximize a specified performance metric as discussed in the next section. Figure 3 is a flow chart of the decision-making process of a PID controller.

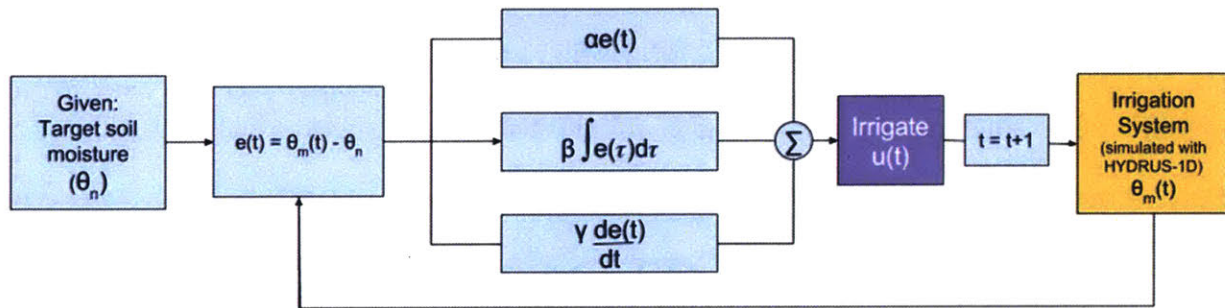


Figure 3: PID Controller Decision Chart

Methods

This section outlines the steps taken to find optimal PID parameters for minimizing irrigation water used while maintaining soil moisture at a level suitable to insure satisfactory yield. First, historical field data were collected and code was written to read and write files that communicate between HYDRUS-1D, MATLAB, and Python. Then, using HYDRUS-1D's built-in bang-bang controller and the field data, a nominal irrigation schedule was determined to minimize the violations of the target soil moisture. Finally, the real-time PID controller parameters were

optimized and tested for accuracy on data sets outside of the one used to optimize the parameters.

Experiment

Historical Data Collection

Nebraska is one of the top five corn growing states in the United States. In southeastern Nebraska, corn makes up most of the acres of harvested cropland (Figure 4). In the counties of the study area (Clay, Fillmore, Gage, Hamilton, Jefferson, Lancaster, Nuckolls, Polk, Saline, Seward, Thayer and York), the acres of corn harvested for grain as a percentage of total harvested area is 47%, over half of which is irrigated (see Appendix 1, p. 36).

In this region summer rainfall is irregular (Figure 5) and not adequate to maintain soil moisture necessary for healthy corn (Garcia y Garcia et al. 2009; Geerts and Raes 2009). According to studies of regional climate variation, the Nebraska is likely to experience changes in available water due to climate change (Liu 2015; Mehta et al. 2015; Scanlon et al. 2012).

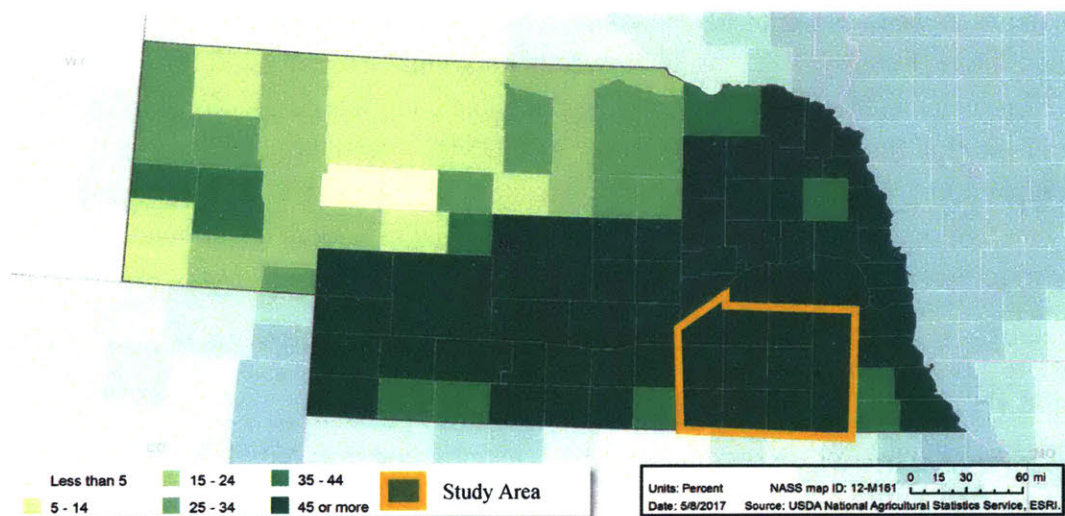


Figure 4:

Acres of Corn Harvested for Grain as a Percent of Harvested Cropland Acreage in Nebraska in 2012

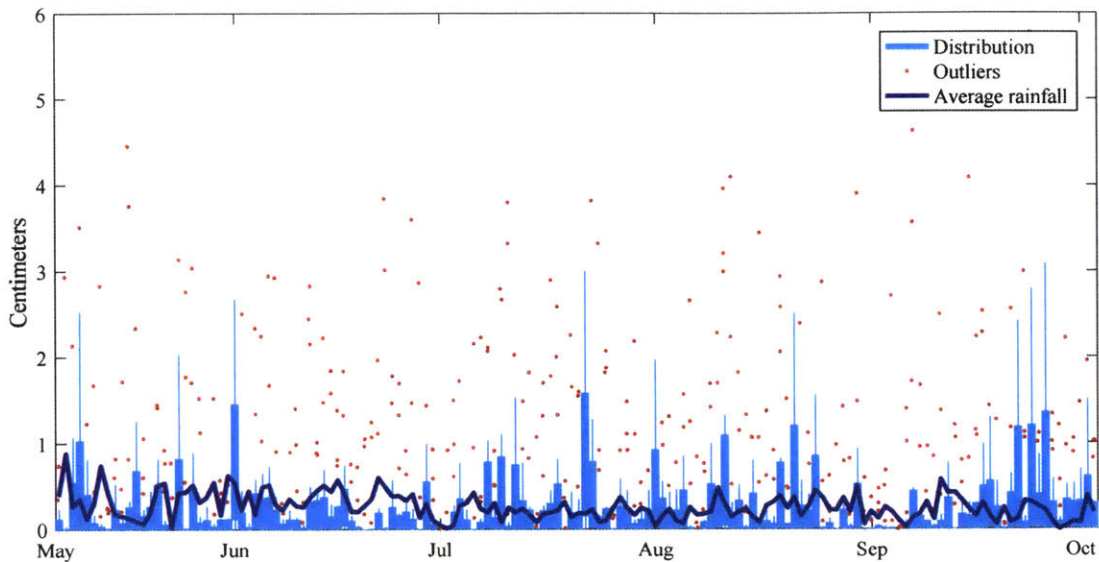


Figure 5: Rainfall for 2000-2015 corn growing season (May 5 – October 1)

These data are used in the HYDRUS-1D model to simulate soil moisture. HYDRUS-1D requires rainfall and reference evapotranspiration data, which were retrieved from the High Plains Regional Climate Center (HPRCC). Their weather stations cover all of Nebraska and the surrounding regions (see Appendix 1, p. 35). The most complete dataset for rainfall and evapotranspiration for the region of interest was 1995 to 2015. Estimates of crop evapotranspiration came from the Food and Agriculture Organization’s crop coefficients for maize. This aggregation of data was used in the HYDRUS-1D model to simulate soil moisture to test the irrigation controllers.

HYDRUS-1D Set-up

HYDRUS-1D requires soil depth and hydraulic properties, season length, a hydraulic model, boundary conditions, a root water uptake model, evaporation, transpiration, and rainfall for all time steps, and a starting matric head for all points in the soil column. The initial values of these parameters are 3 meters of soil depth to accommodate the 1 meter root growth of corn, a season of 150 days (the average season length for summer corn, the season starts May 5 and ends October 1), evapotranspiration and rainfall from the meteorological data gathered from the HPRCC, and starting soil moisture is field capacity (-73 cm). The soil hydraulic model is van Genuchten-Mualem with no hysteresis (Mualem 1976). The soil hydraulic parameters are those for “sandy loam” built into the HYDRUS database (van Genuchten-Mualem parameters of $\theta_s = 0.41$, $\theta_r = 0.065$, $\alpha = 0.075 \text{ cm}^{-1}$, $n = 1.89$, $K_s = 106.1 \text{ cm/day}$, and $l = 0.5$). The upper boundary condition is “atmospheric with surface runoff” to remove ponding on the surface and the lower boundary condition is “free drainage” since the water table is unlikely to be in the root zone this model does not account for waterlogging. The root water uptake model is from Feddes

et al. 1978 and uses the parameters for “Wesseling 1991 Corn.” Finally, the observation node that acts as a surrogate soil moisture sensor for the irrigation controllers is placed at a depth of 10 cm from the top of the soil. A depth of 10 cm was chosen to align with drip irrigation placement studies that focus on depths of 0 to 15 and 15 to 30 cm to study the plant-available water (Thongsaga and Ranamukhaarachchi 2009). This set-up assumes that the plant is exclusively affected by the climate, not nutrient availability or toxicity.

Code Structure

In order to run the optimizations, the code structure incorporated Python, MATLAB, and HYDRUS-1D. Python was used to read and write the text files required to run and get information from HYDRUS-1D (see Appendix 2, p. 37). MATLAB housed the optimization; all of the following optimizations were found using the `fmincon` function (see Appendix 3, p. 47). `Fmincon` is designed to find the minimum of a constrained nonlinear multivariable function.

Finding nominal irrigation – Bang-Bang Controller Optimization

A PID controller requires nominal soil moisture and irrigation rate targets for the controller to use as baselines. The nominal irrigation rate was determined using the triggered irrigation built into HYDRUS-1D and optimized to maintain a target soil moisture for the entire season.

The nominal irrigation rate was determined using the triggered irrigation algorithm (a bang-bang controller dependent on soil moisture) built into HYDRUS-1D. This algorithm is designed to maintain a target soil moisture for the entire season. To compute the triggered irrigation rate in HYDRUS, values are required for the matric head that triggers irrigation, the irrigation rate, the irrigation duration, and the lag time. This is similar to most bang-bang controllers that are widely commercially available. Based on the literature, the target volumetric soil moisture should be somewhere between 0.2 and 0.7 and the evapotranspiration should be exactly matched by irrigation for maximum yield (Huang 2004; Kanemasu et al. 1983; Payero et al. 2008, 2009; Rivera-Hernandez et al. 2010). For this analysis, the target volumetric soil moisture was assumed to be 0.2 – a pressure head of -35.4081 cm – for the entire season. The lag time was set to 0 days and the other two parameters – irrigation duration and amount – were the decision variables for optimization in MATLAB. The daily average evapotranspiration was calculated for the counties in the study area for the entire 150 day growing season from the reference evapotranspiration data at 22 stations in the region (see Appendix 1, p. 35). The reference evapotranspiration data was multiplied by the crop factors (K_c) from FAO to derive crop transpiration (see Appendix 1, p. 35). There was no rainfall input in this system. The objective of this optimization was to minimize the total amount below the target soil moisture over a season. The following equations are the optimization objective function and constraints given to `fmincon`.

Objective function: minimize the sum of the negative values of $\theta_m - \theta_n$ by changing irrigation duration and amount

Constraints:

- $10 \text{ minutes} \leq \text{irrigation duration} \leq 1 \text{ day}$
- $0.1 \text{ cm} \leq \text{irrigation amount} \leq 10 \text{ cm}$

In the first few trials, the irrigation duration and amount optimization results were highly dependent on initial conditions. For example, a starting pair of 0.4 days of irrigation and 5 cm/day of irrigation at each time the trigger is reached returned an “optimized” pair of 0.398 days and 5 cm/day of irrigation or return the initial values stating that they presented a local minimum that satisfied the constraints. This indicated that there is no global minimum for this problem. A complete, exhaustive evaluation of the irrigation duration from 0 to 1 day and irrigation amount from 0.1 to 10 cm/day illustrates this fact (Figure 6).

Figure 6 illustrates the difficulty of finding a true minimum when most of the values of the parameters provide equally low objective values. All values in the region where irrigation amount is larger than 4 cm and irrigation duration ranges from 0.2 to 0.8 days are equally valid as nominal irrigation values for the optimization. 5 cm/day of irrigation for 0.5 days was used as the nominal triggered irrigation for the following procedures because it is a local minimum in the region that minimizes the target soil moisture violation.

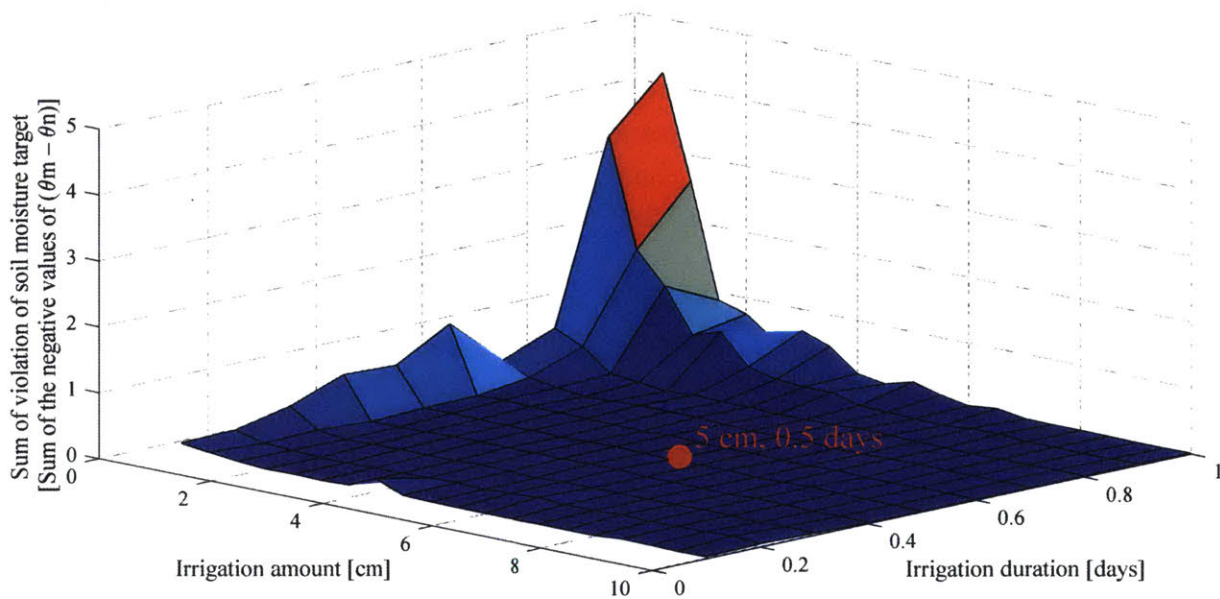


Figure 6: Exhaustive Evaluation of Objective Function to Minimize Violation of Target Soil Moisture

The 30-day smoothed average over the years 2000-2015 serves as the nominal irrigation for the real-time PID controller (Figure 7). The orange line (smoothed average) is used as the nominal irrigation for the PID optimization instead of the rapidly changing average irrigation to make it easier for the PID to track the deviations from the nominal without large variations in the derivative component.

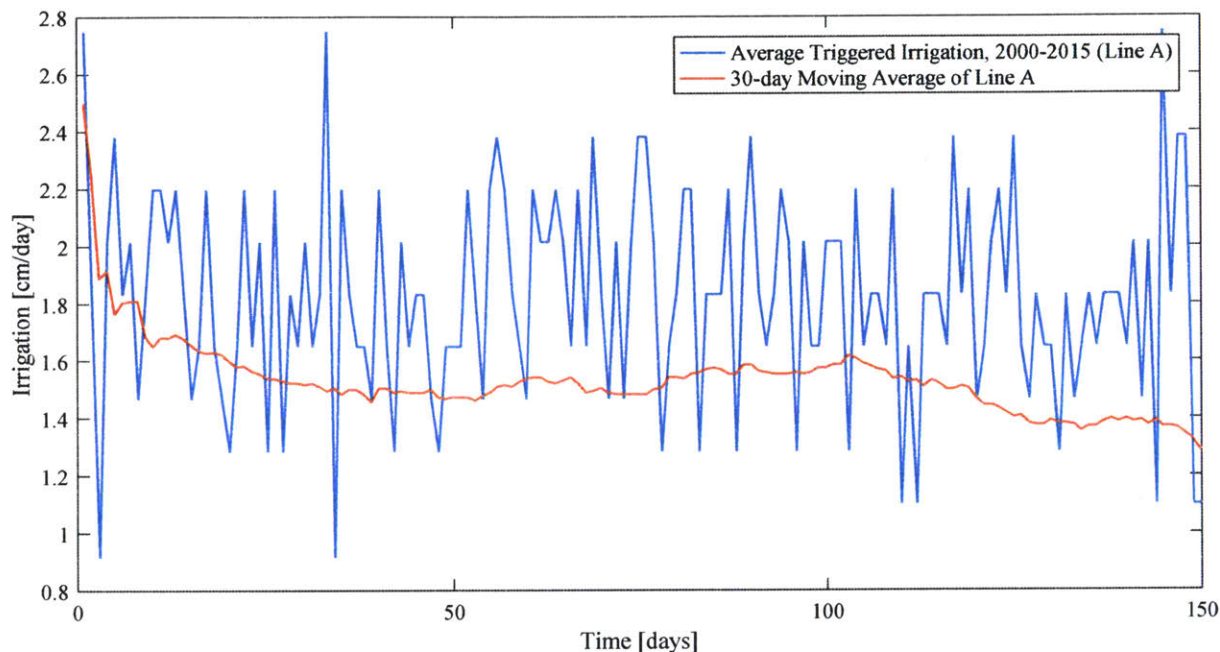


Figure 7: Average of Triggered Irrigation to be used as Nominal Irrigation for PID

Finding Optimal PID Parameters

The optimization of the parameters for the PID controller builds on the previous steps and applies PID control logic to the information we have collected. The PID controller is designed to minimize deviations from a specified target. Here the PID target values are the 30-day moving average irrigation rate from the bang-bang controller (Figure 7) and the target soil moisture (0.2) for all 150 days. The equation for a generic PID controller can be rewritten with:

$$\begin{aligned}
 u(t) &= dQ = Q(t) - Q(\text{nominal}) & \text{Equation 7: Values for PID for Irrigation} \\
 e(t) &= d\theta = \theta_m(t-1) - \theta_n & \text{Optimization}
 \end{aligned}$$

Where Q is irrigation rate [cm/day] and θ is volumetric soil moisture [·].

The optimization objective function took parameters α , β , γ and used them in a PID controller to minimize total irrigation. The PID controller used the same starting conditions as the season-long triggered irrigation method, but stepped through the simulation one day at a time for the entire 150-day season measuring soil moisture from HYDRUS at the end of each day to use in the next decision. The integral was taken over a 5-day moving window to account for the most recent

rainfall event but ‘forget’ the dry periods early in the summer. This is the same order of magnitude as the time for gravity drainage of the soil.

Objective function: minimize total water applied over 150 days using a real-time PID controller by varying parameters α , β , and γ .

The parameters α and γ never moved very far, if at all, from their initial condition. But β varied for different α and γ parameters. The reason for this is similar to the issue with minimizing triggered irrigation parameters. There is no global minimum in this highly nonlinear system, only local minimums. These local minimums seem to only matter when β changes, not α or γ . Figure 8 and Figure 9 illustrate why this is the case, there is only slight improvement in irrigation water use and no improvement in soil moisture maintenance when decreasing α . On the other hand, there is a dramatic increase in the amount of time the soil violates its target soil moisture as β increases and there is a corresponding increase in total irrigation water use with small β values. Beta represents the fundamental tension in the system – maintaining soil moisture uses more water than allowing soil moisture depletion. These trends are similarly reflected in the comparison of γ and β .

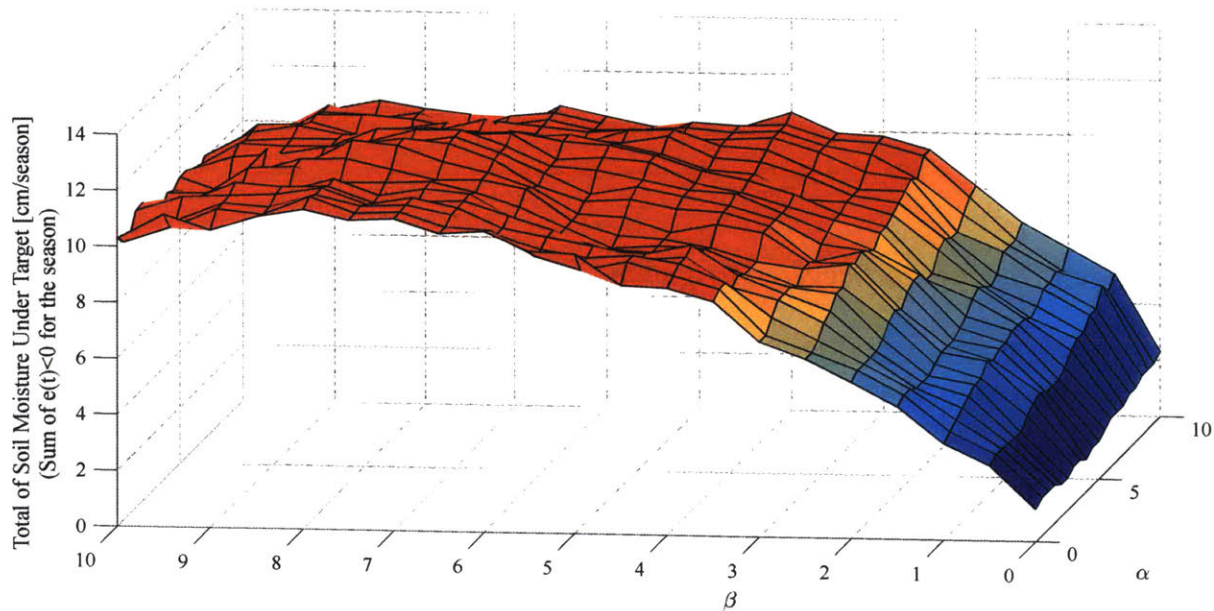


Figure 8: Total Violations of Soil Moisture Target over 150 day season, variations with α and β

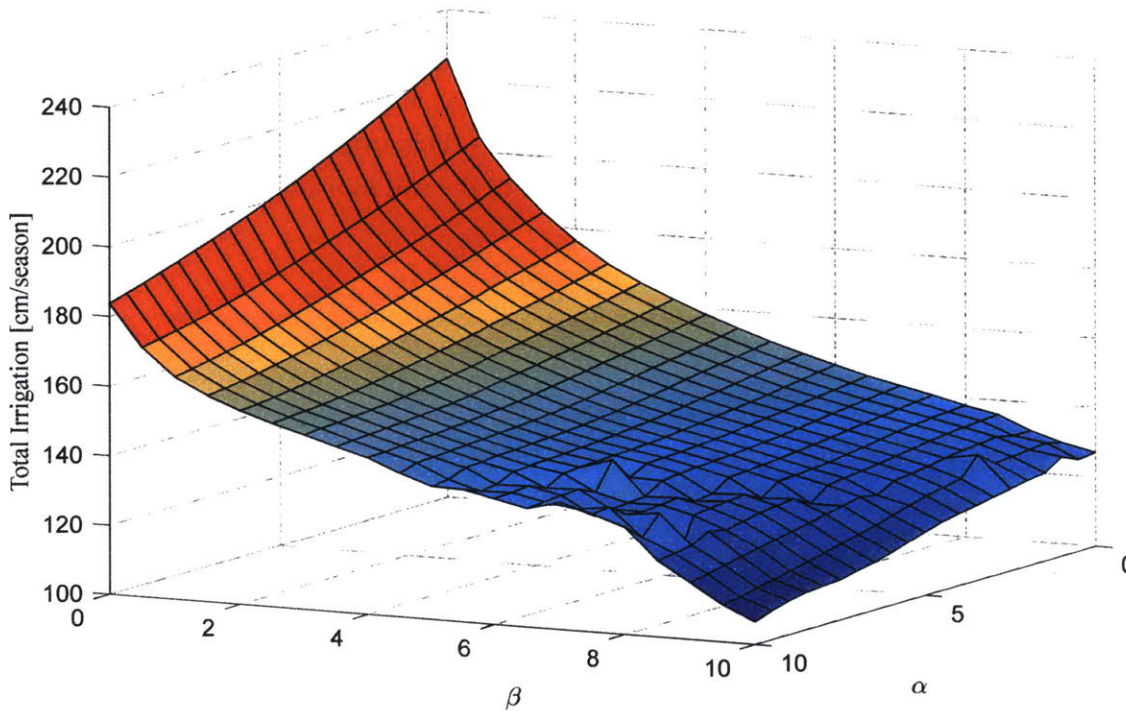


Figure 9: Total irrigation over a 150 day season, variations with α and β

Ultimately, β is the only parameter that will dramatically change the system. The local minimum of the optimization depends heavily on the chosen α and γ values.

Results

The parameters for a PID controller did not yield results that provide a clear answer on how to optimize the system as β is the only parameter that changes to find a local minimum. However, the real-time PID controller (although perhaps not “ideal”) did apply less water than the real-time bang-bang controller. Since we know that HYDRUS-1D is analogous to scheduling irrigation, it is safe to say that PID controllers could be the next step forward in low-cost, easy to implement irrigation control systems.

shows a comparison of the real-time bang-bang irrigation response from HYDRUS-1D, the nominal irrigation for the PID, and the optimal real-time PID irrigation for the year 2000.

The real-time bang-bang irrigation response is based solely on the closed-loop bang-bang controller from HYDRUS-1D with a predetermined irrigation amount of 5 cm/day for 0.5 days when the soil moisture target is not met. This strategy uses 185 cm of water for the season and

never violates the soil moisture target. The nominal irrigation for the PID is the 30-day smoothed average of the 15 year average of all the bang-bang controlled seasons from 2000-2015. This irrigation strategy used 228.4 cm of water for the season and only once violates the soil moisture target. The optimal real-time PID irrigation is based on chosen α and γ values – 4 and 8 respectively. This returned an optimal value of 2.99 for β . This irrigation method uses 150.9 cm of water for the season and shows multiple violations of the soil moisture target.

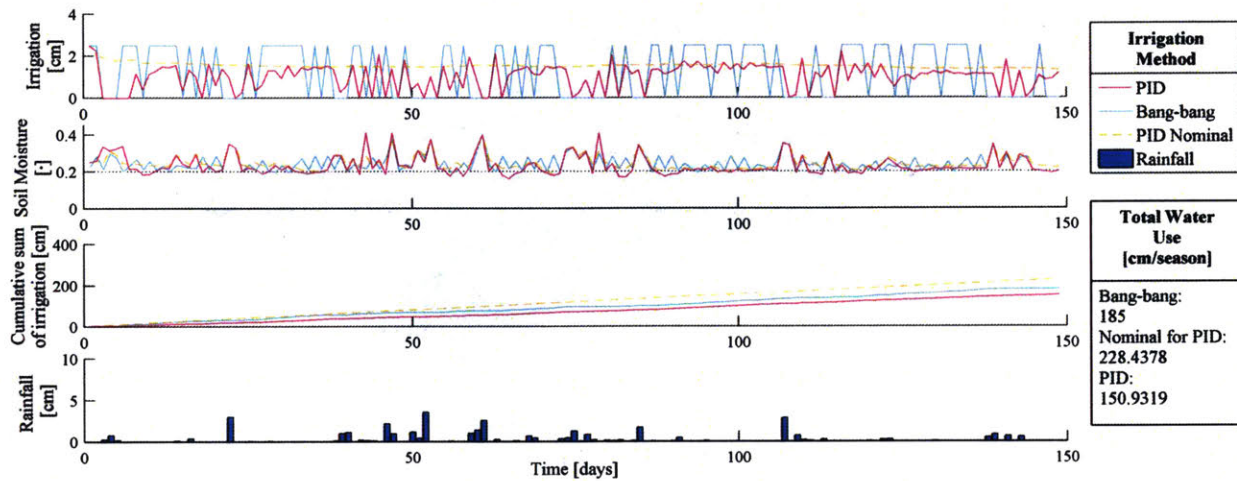


Figure 9: Irrigation for 2000

The PID controller from the 2000 optimization, in which $\alpha = 4$, $\beta = 2.99$, and $\gamma = 8$, was applied to 5 test years, 1995 – 1999. The following graphs (Figures 10-14) show that the optimal PID controller always uses the least water for the season, but violates the target soil moisture at least once per season, which could endanger crop health. The PID controller performs better than the triggered irrigation during storms, adding considerably less water than the bang-bang controller when it rains. Additionally, the graph of PID irrigation rate is much smaller than the bang-bang in all years, illustrating the need for precision irrigation as the volumes of water are much less than traditional sprinklers can provide.

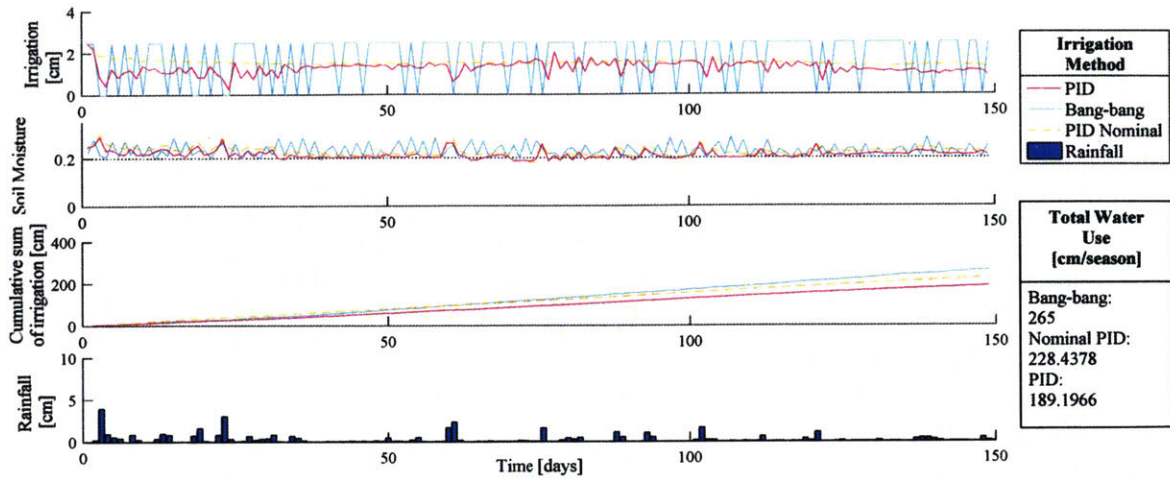


Figure 10: 1995 Irrigation

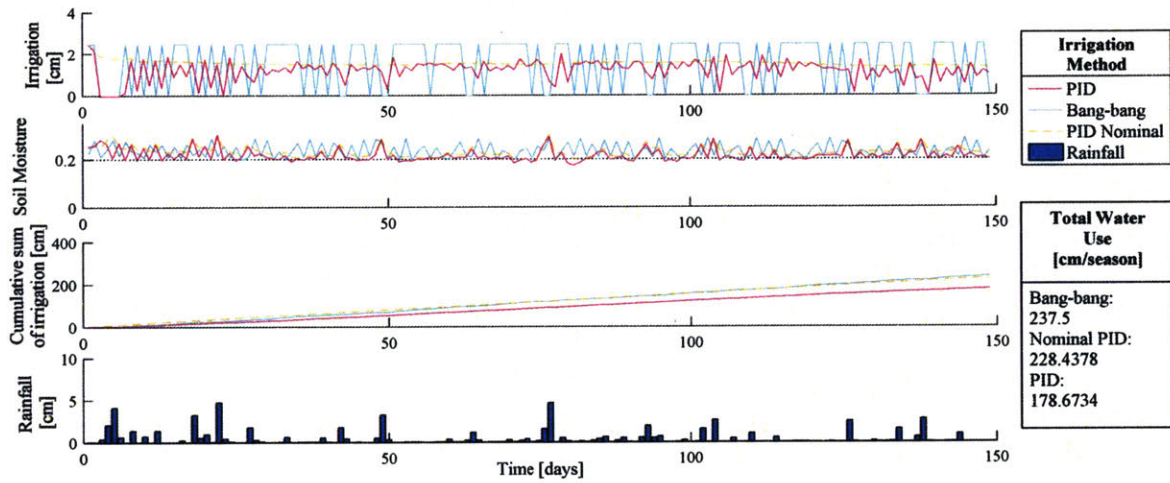


Figure 11: 1996 Irrigation

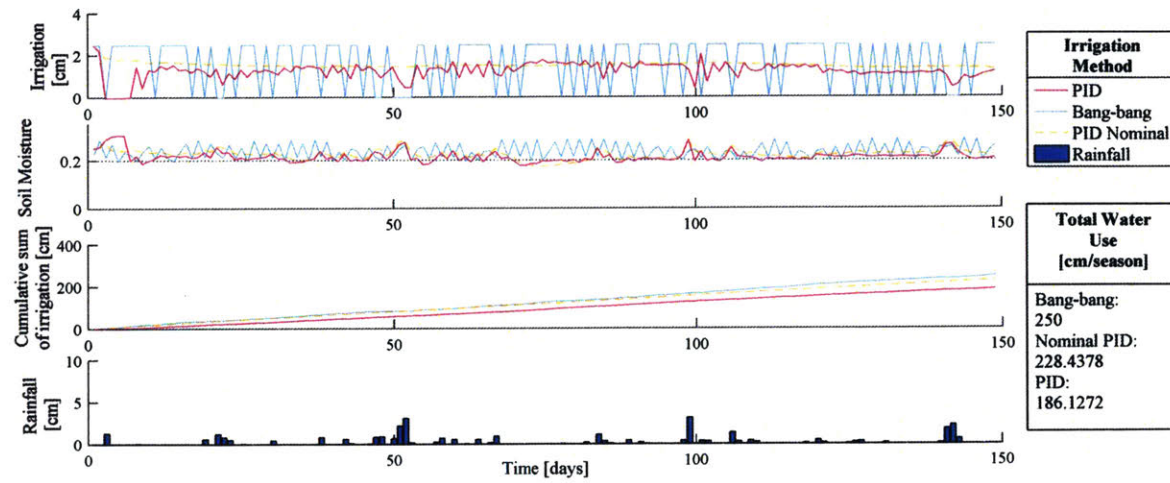


Figure 12: 1997 Irrigation

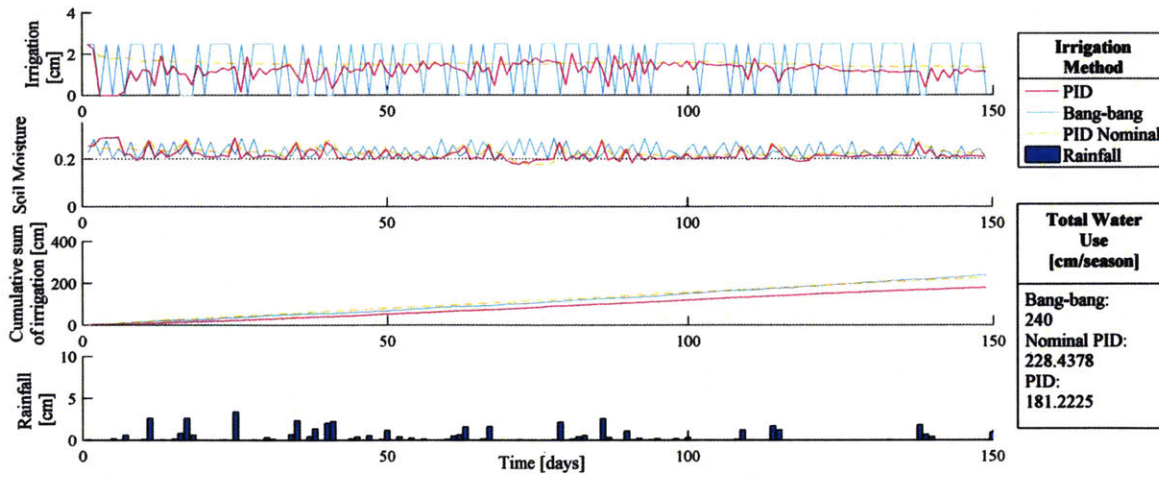


Figure 13: 1998 Irrigation

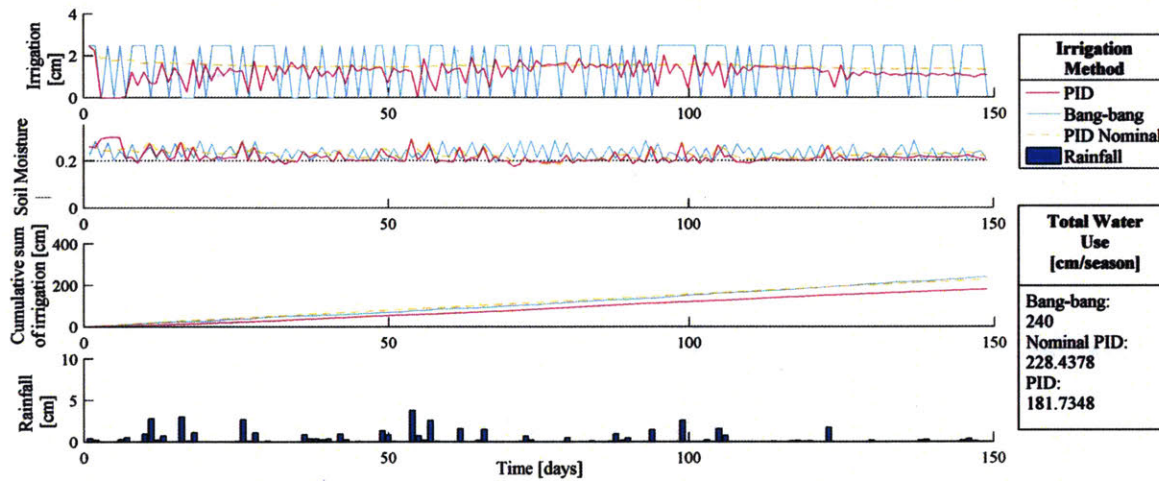


Figure 14: 1999 Irrigation

Discussion

From the literature, we know that bang-bang controllers reduce water use as opposed to traditional open-loop control systems. The PID controller shows promise for further reduction of water use. The PID controller uses less water, but also violates the minimum soil moisture constraint, which the bang-bang controller does not. Changing the minimum soil moisture over the season to a slightly higher value than the minimum allowable would be a good way to resolve this issue (Payero et al. 2009). Irrigation control parameters must be calibrated to the installation location. In regions where there are irregular rain events, this thesis shows that PID parameter optimization can be done off-line with HYDRUS-1D and historical weather data.

The sole dependence of the PID controller optimization on β can be explained by two things. First, changing α and γ may not make an impact physically. Increasing α will increase the PID's sensitivity to the error term in the previous time-step. Because soil has a natural memory, it does not fill up or dry out very quickly. This soil, including crop water uptake, took 3 days to dry out completely. The proportional term only accounts for the most recent time step (24 hours). Therefore, α has little role to play in adding water to the system. Similarly, γ depends on the difference between the two most recent errors. It is unlikely that the error on day 3 will be much higher than on day 2 because the soil does not dry on this time scale. So, α and γ would only be helpful in 'removing' water after a large rainfall event. In a physical system, water cannot be rapidly removed from the soil and therefore these two parameters are not very helpful in optimizing the PID. Second, the interval of the integration term is helpful for PID optimization. Initially, the optimization was built to integrate over all previous errors from the start of the season. This resulted in stalling of the optimization solver. Once the interval was shortened to 5 days, on the order of magnitude of gravity drainage of the soil column, the integral term became more meaningful. Therefore, it makes physical sense that β is the sole parameter to optimize in a PID system that is measuring soil water content at a 1 day interval.

It should be noted that there were many instances of HYDRUS-1D failing to initialize in the optimization process. If this occurred on the initial condition, the optimization would stall. However, if this occurred in one of the iterations, MATLAB `fmincon` occasionally stalls but other times it found a local minimum regardless. This is heavily dependent on the starting conditions of the field and the initial input parameters. This indicates that this optimization is not robust enough to be implemented in the field immediately.

In the future, this work could be expanded to include minimization of nutrient runoff or minimization of salt build-up in the root zone using the tools developed in this thesis. Changing the target soil moisture based on a simple crop model is another possible development.

Conclusion

This thesis contributes to the existing literature on soil moisture based irrigation controllers. It describes a procedure for using real-time PID controllers to minimize water use with HYDRUS-1D, an accurate model to simulate soil moisture for irrigation control. In a comparison of two closed-loop irrigation controllers: bang-bang and PID for minimizing water use, the PID controller uses less water. However, the PID controller violates the minimum soil moisture constraint at least once per season, which could endanger plant health (Geerts and Raes 2009). This problem could be fixed by changing the soil moisture targets for the controller. Ultimately, PID controllers offer a mid-point between the simplistic bang-bang controllers that are widely used and the model based controllers that require large datasets, wireless network infrastructure, and robust computing systems. PID controllers can be implemented in the field with the same sensors that are widely used with bang-bang controllers today resulting in a reduction of water use in regions where water is scarce or availability is irregular.

Bibliography

- Adeyemi, O., Grove, I., Peets, S., and Norton, T. (2017). "Advanced Monitoring and Management Systems for Improving Sustainability in Precision Irrigation." *Sustainability*, 9(353), 1–29.
- Aguilar, J. V., Langarita, P., Rodellar, J., Linares, L., and Horváth, K. (2016). "Predictive control of irrigation canals ??? robust design and real-time implementation." *Water Resources Management*, Water Resources Management, 30(11), 3829–3843.
- Akhtar, F., Tischbein, B., and Awan, U. K. (2013). "Optimizing Deficit Irrigation Scheduling Under Shallow Groundwater Conditions in Lower Reaches of Amu Darya River Basin." *Water Resources Management*, 27(8), 3165–3178.
- Alexandratos, N., and Bruinsma, J. (2012). "World agriculture towards 2015/2030: The 2012 Revision." *ESA Working Paper*, No. 12-03(12), 147.
- Álvarez, A., Ridao, M., Ramirez, D., and Sánchez, L. (2013). "Constrained Predictive Control of an Irrigation Canal." *Journal of Irrigation and Drainage Engineering*, 139(10), 841–854.
- Bahat, M., Inbar, G., Yaniv, O., and Schneider, M. (2000). "A Fuzzy irrigation controller system." *Engineering Applications of Artificial Intelligence*, 13(2), 137–145.
- Bi, P., and Zheng, J. (2014). "Study on application of grey prediction fuzzy PID control in water and fertilizer precision irrigation." *Proceedings - 2014 IEEE International Conference on Computer and Information Technology, CIT 2014*, (0), 789–791.
- Bolea, Y., Puig, V., and Blesa, J. (2014a). "Linear parameter varying modeling and identification for real-time control of open-flow irrigation canals." *Environmental Modelling and Software*, Elsevier Ltd, 53, 87–97.
- Bolea, Y., Puig, V., and Blesa, J. (2014b). "Gain-Scheduled Smith Predictor PID-Based LPV Controller for Open-Flow Canal Control." *Ieee Transactions on Control Systems Technology*, 22(2), 468–477.
- Cardenas-Lailhacar, B., and Dukes, M. D. (2010). "Precision of soil moisture sensor irrigation controllers under field conditions." *Agricultural Water Management*, Elsevier B.V., 97(5), 666–672.
- Dabach, S., Lazarovitch, N., Šimůnek, J., and Shani, U. (2013). "Numerical investigation of irrigation scheduling based on soil water status." *Irrigation Science*, 31(1), 27–36.
- Dobriyal, P., Qureshi, A., Badola, R., and Hussain, S. A. (2012). "A review of the methods available for estimating soil moisture and its implications for water resource management." *Journal of Hydrology*, Elsevier B.V., 458–459, 110–117.
- Evans, R. G., LaRue, J., Stone, K. C., and King, B. A. (2013). "Adoption of site-specific variable rate sprinkler irrigation systems." *Irrigation Science*, 31(4), 871–887.
- Evelt, S., and Howell, T. (2000). "Automatic drip irrigation of corn and soybean." *Proceedings of the 4th Decennial National Irrigation Symposium*, (1993), 401–408.
- FAO. (2017). "Maize Crop Information." <<http://faostat3.fao.org/faostat-gateway/go/to/home/E>> (Jul. 5, 2017).
- Feddes, R. A., Kowalik, P. J., and Zaradny, H. (1978). *Simulation of Field Water Use and Crop Yield*.
- Garcia y Garcia, A., Guerra, L. C., and Hoogenboom, G. (2009). "Water use and water use efficiency of sweet corn under different weather conditions and soil moisture regimes."

- Agricultural Water Management*, 96(10), 1369–1376.
- Gebbers, R., and Adamchuk, V. I. (2010). “Precision Agriculture and Food Security.” *Science*, 327(5967), 828–831.
- Geerts, S., and Raes, D. (2009). “Deficit irrigation as an on-farm strategy to maximize crop water productivity in dry areas.” *Agricultural Water Management*, 96(9), 1275–1284.
- van Genuchten, M. T. (1980). “A Closed-form Equation for Predicting the Hydraulic Conductivity of Unsaturated Soils.” *Soil Science Society of America Journal*, 44(5), 892.
- Giusti, E., and Marsili-Libelli, S. (2015). “A Fuzzy Decision Support System for irrigation and water conservation in agriculture.” *Environmental Modelling & Software*, Elsevier Ltd, 63, 73–86.
- Greenwood, D. J., Zhang, K., Hilton, H. W., and Thompson, A. J. (2010). “Opportunities for improving irrigation efficiency with quantitative models, soil water sensors and wireless technology.” *Journal of Agricultural Science*, 148(1), 1–16.
- Huang, G. (2004). “Modeling soil water regime and corn yields considering climatic uncertainty.” *Plant and Soil*, 259(1–2), 221–229.
- Jones, H. G. (2004). “Irrigation scheduling: Advantages and pitfalls of plant-based methods.” *Journal of Experimental Botany*, 55(407), 2427–2436.
- Kanemasu, E. T., Steiner, J. L., Biere, A. W., Worman, F. D., and Stone, J. F. (1983). “Irrigation in the Great Plains.” *Agricultural Water Management*, 7(1–3), 157–178.
- Kolassa, J., Aires, F., Polcher, J., Prigent, C., Jimenez, C., and Pereira, J. M. (2013). “Soil moisture retrieval from multi-instrument observations: Information content analysis and retrieval methodology.” *Journal of Geophysical Research Atmospheres*, 118(10), 4847–4859.
- Lacasta, A., Morales-Hernández, M., Brufau, P., and García-Navarro, P. (2014). “Simulation of PID control applied to irrigation channels.” *Procedia Engineering*, 70, 978–987.
- Liu, H. P. and B. R. S. and Y. S. and R. C. R. and D. L. and C. (2015). “Impacts of varying agricultural intensification on crop yield and groundwater resources: comparison of the North China Plain and US High Plains.” *Environmental Research Letters*, IOP Publishing, 10(4), 44013.
- Lozano, D., Arranja, C., Rijo, M., and Mateos, L. (2010). “Simulation of automatic control of an irrigation canal.” *Agricultural Water Management*, 97(1), 91–100.
- Mailhol, J. C., Ruelle, P., Walser, S., Schütze, N., and Dejean, C. (2011). “Analysis of AET and yield predictions under surface and buried drip irrigation systems using the Crop Model PILOTE and Hydrus-2D.” *Agricultural Water Management*, Elsevier B.V., 98(6), 1033–1044.
- McBratney, A., Whelan, B., Ancev, T., and Bouma, J. (2005). “Future Directions of Precision Agriculture.” *Precision Agriculture*, 6(July 2004), 7–23.
- McCarthy, A. C., Hancock, N. H., and Raine, S. R. (2013). “Advanced process control of irrigation: The current state and an analysis to aid future development.” *Irrigation Science*, 31(3), 183–192.
- McCarthy, A. C., Hancock, N. H., and Raine, S. R. (2014). “Simulation of Irrigation Control Strategies for cotton using Model Predictive Control within the VARIwise simulation framework.” *Computers and Electronics in Agriculture*, 101, 135–147.
- McClendon, R. W., Hoogenboom, G., and Seginer, I. (1996). “Optimal control and neural networks applied to peanut irrigation management.” *Transactions of the ASAE*, 39(1), 275–

- Mehta, V. M., Mendoza, K., Daggupati, P., Srinivasan, R., Rosenberg, N. J., and Deb, D. (2015). "High-resolution Simulations of Decadal Climate Variability Impacts on Water Yield in the Missouri River Basin with the Soil and Water Assessment Tool (SWAT)." *Journal of Hydrometeorology*, 151104123251006.
- Mualem, Y. (1976). "A new model for predicting the hydraulic conductivity of unsaturated porous media." *Water Resources Research*, 12(3), 513–522.
- Muñoz-Carpena, R., Dukes, M. D., Li, Y., and Klassen, W. (2008). "Design and field evaluation of a new controller for soil-water based irrigation." *Applied Engineering in Agriculture*, 24(2), 183–191.
- Park, M. H., Stenstrom, M., and Pincetl, S. (2009). "Water quality improvement policies: Lessons learned from the implementation of proposition O in Los Angeles, California." *Environmental Management*, 43(3), 514–522.
- Payero, J. O., Tarkalson, D. D., Irmak, S., Davison, D., and Petersen, J. L. (2008). "Effect of irrigation amounts applied with subsurface drip irrigation on corn evapotranspiration, yield, water use efficiency, and dry matter production in a semiarid climate." *Agricultural Water Management*, 95(8), 895–908.
- Payero, J. O., Tarkalson, D. D., Irmak, S., Davison, D., and Petersen, J. L. (2009). "Effect of timing of a deficit-irrigation allocation on corn evapotranspiration, yield, water use efficiency and dry mass." *Agricultural Water Management*, 96(10), 1387–1397.
- Pereira, L. S., Oweis, T., and Zairi, A. (2002). "Irrigation management under water scarcity." *Agricultural Water Management*, 57(3), 175–206.
- Protopapas, A. L., and Georgakakos, A. P. (1990). "An optimal control method for real-time irrigation scheduling." 26; 26(4), 647–699.
- Richards, L. A. (1931). "Capillary conduction of liquids through porous mediums." *Journal of Applied Physics*, 1(5), 318–333.
- Rivera-Hernandez, B., Carrillo-Avila, E., Obrador-Olan, J. J., Juarez-Lopez, J. F., and Aceves-Navarro, L. A. (2010). "Morphological quality of sweet corn (*Zea mays* L.) ears as response to soil moisture tension and phosphate fertilization in Campeche, Mexico." *Agricultural Water Management*, 97(9), 1365–1374.
- Romero, R., Muriel, J. L., García, I., and Muñoz de la Peña, D. (2012). "Research on automatic irrigation control: State of the art and recent results." *Agricultural Water Management*, 114, 59–66.
- Rosegrant, M. W., Cai, X., and Cline, S. (2002). *World Water and Food to 2025: Dealing with Scarcity. Food Policy*.
- Sammis, T., Sharma, P., Shukla, M. K., Wang, J., and Miller, D. (2012). "A water-balance drip-irrigation scheduling model." *Agricultural Water Management*, Elsevier B.V., 113, 30–37.
- Sarwar, A., Bastiaanssen, W. G. M., and Feddes, R. A. (2001). "Irrigation water distribution and long-term effects on crop and environment." 50, 125–140.
- Scanlon, B. R., Faunt, C. C., Longuevergne, L., Reedy, R. C., Alley, W. M., McGuire, V. L., and McMahon, P. B. (2012). "Groundwater depletion and sustainability of irrigation in the US High Plains and Central Valley." *Proceedings of the National Academy of Sciences of the United States of America*, 109(24), 9320–5.
- Seidel, S. J. (2015). "Optimal Irrigation Scheduling, Irrigation Control and Drip Line Layout to Increase Water Productivity and Profit in Subsurface Drip-Irrigated Agriculture." *Irrigation and Drainage*, 64(4), 501–518.

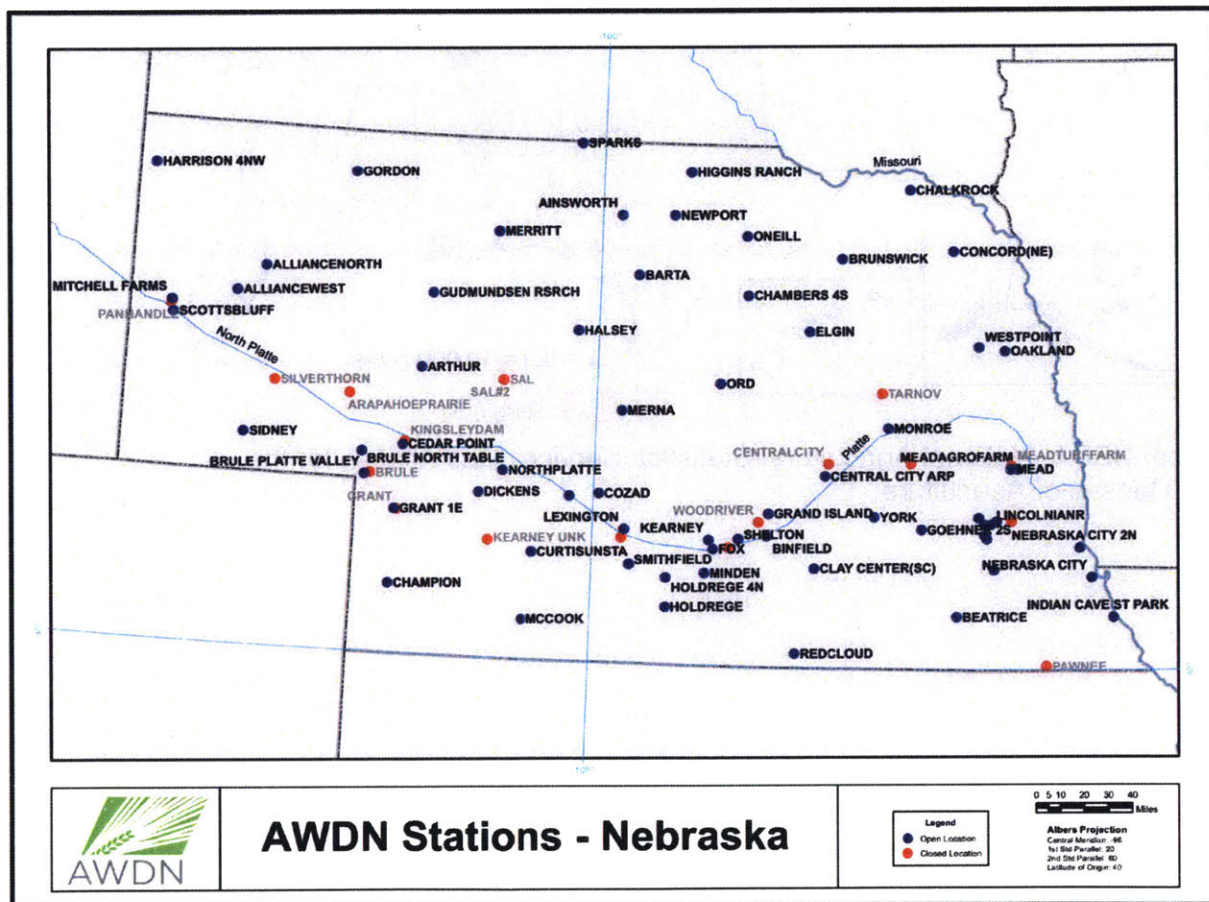
- Shani, U., Tsur, Y., and Zemel, A. (2004). "Optimal dynamic irrigation schemes." *Optimal Control Applications and Methods*, 25(2), 91–106.
- Šimůnek, J., Šejna, M., Saito, H., Sakai, M., and Genuchten, M. T. van. (2009). "HYDRUS-1D Software Package."
- Soulis, K. X., Elmaloglou, S., and Dercas, N. (2015). "Investigating the effects of soil moisture sensors positioning and accuracy on soil moisture based drip irrigation scheduling systems." *Agricultural Water Management*, Elsevier B.V., 148, 258–268.
- Stafford, J. V. (2000). "Implementing Precision Agriculture in the 21st Century." *Journal of Agricultural Engineering Research*, 76(3), 267–275.
- Tafteh, A., and Sepaskhah, A. R. (2012). "Application of HYDRUS-1D model for simulating water and nitrate leaching from continuous and alternate furrow irrigated rapeseed and maize fields." *Agricultural Water Management*, Elsevier B.V., 113, 19–29.
- Tan, X., Shao, D., and Liu, H. (2014). "Simulating soil water regime in lowland paddy fields under different water managements using HYDRUS-1D." *Agricultural Water Management*, Elsevier B.V., 132, 69–78.
- Thongsaga, K., and Ranamukhaarachchi, S. L. (2009). "Simulation of Growth and Yield of Maize under Water Stress Imposed during Critical Growth Periods in." *Asia-Pacific Journal of Rural Development*, XIX(1), 109–135.
- Wang, X., Huang, G., Yang, J., Huang, Q., Liu, H., and Yu, L. (2015). "An assessment of irrigation practices: Sprinkler irrigation of winter wheat in the North China Plain." *Agricultural Water Management*, Elsevier B.V., 159, 197–208.
- Xiang, X. (2011). "Design of Fuzzy Drip Irrigation Control System Based on ZigBee Wireless Sensor Network." 495–501.
- Zazueta, F. S., and Xin, J. (1994). "Soil Moisture Sensors." *Measurement*, (April), 1–11.
- Zotarelli, L., Dukes, M. D., Scholberg, J. M. S., Femminella, K., and Muñoz-Carpena, R. (2011). "Irrigation Scheduling for Green Bell Peppers Using Capacitance Soil Moisture Sensors." *Journal of Irrigation and Drainage Engineering*, 137(February), 73–81.

Appendix 1: Data Collection

FAO Crop Factors

The crop factor (Kc) relating water requirements to reference evapotranspiration for different crop growth stages of grain maize is for the initial stage 0.3-0.5 (15 to 30 days), the development stage 0.7-0.85 (30 to 45 days) the mid-season stage 1.05-1.2 (30 to 45 days), during the late season stage 0.8-0.9 (10 to 30 days), and at harvest 0.55-0.6 (FAO 2017).

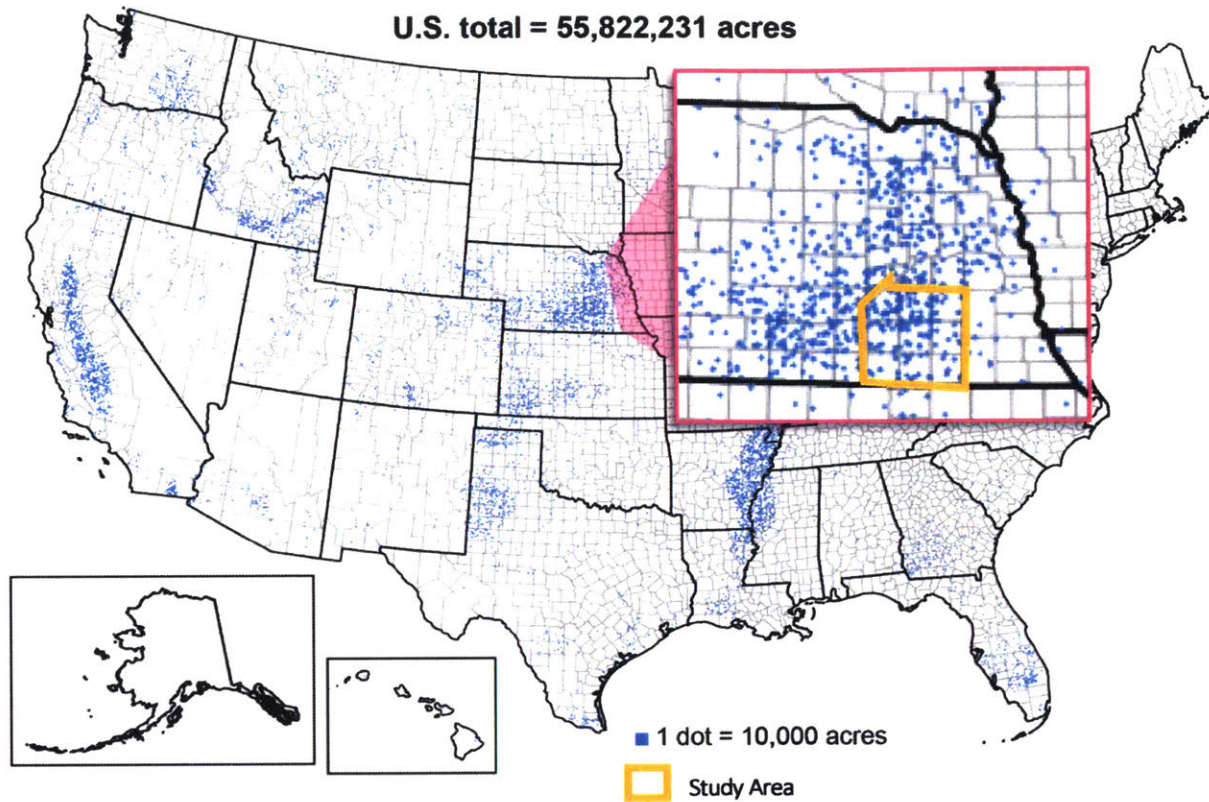
High Plains Regional Climate Center Stations



Selected Stations: Beatrice, Binfield, Central City, Central City Airport, Grand Island, Indian Cave, Lincoln (9N6E, 10E17N, 12W55N, 20E35S, 27E56S, 51E13S, 82E20S, 93E34S), Lincoln IANR, Mead, Mead AgroFarm, Monroe, Nebraska City, Nebraska City 2, Red Cloud, Shelton, York

Acres of Irrigated Land in 2012

U.S. total = 55,822,231 acres



Source: USDA, National Agricultural Statistics Service, Map Atlases for the 2012 Census of Agriculture.

Appendix 2: Code from Python

Adapted from “Hydrus Parameter File Adaptor” on GitHub as hydrus-wrapper

Project: Phd Meisam Rezaei

Author: Van Hoey Stijn

```
import os
import sys
import time
import datetime
import subprocess
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
from matplotlib import cm
import shutil
import math as math
import scipy.io
```

```
#-----
# INPUT/OUTPUT ROUTINES
#-----
```

```
def replaceInputWater(path_to_dir, newvalue, parname='Ks', layer=1):
```

```
    """
```

```
    The Hydrus input file Selector.in governs the input water, irrigation rate and duration, and
    boundary conditions.
```

```
    The parameters values are given for each profile layer under the parameter
    name. As such, this definition search for the parameter and layer and
    changes the par.
```

```
    Parameters
```

```
    -----
```

```
    path_to_dir:
```

```
        Directory with the Hydrus-input and output files in
```

```
    newvalue:
```

```
        New parameter value to be used, %.9f value
```

parname:

The name of the parameter as is appears in the file

layer:

The layer where the parameter need to be changed

'''

try:

```
os.rename(os.path.join(path_to_dir,'Selector.in'),os.path.join(path_to_dir,'Selector_old.in'))
```

except:

```
os.remove(os.path.join(path_to_dir,'Selector_old.in'))
```

```
os.rename(os.path.join(path_to_dir,'Selector.in'),os.path.join(path_to_dir,'Selector_old.in'))
```

```
with open(os.path.join(path_to_dir,'Selector_old.in'),'r') as fin:
```

```
with open(os.path.join(path_to_dir,'Selector_new.in'),'w') as fout:
```

```
    fintext = fin.readlines()
```

```
    #Get line with par headers assuming Ks is always a parameter
```

```
    #using the parameter is not possible, since eg 'n' would give errors
```

```
    parstartline = fintext.index([x for x in fintext if 'Ks' in x][0])
```

```
    #Get index (column) of the parameter
```

```
    parcolumn = fintext[parstartline].split().index(parname)
```

```
    #adapting the lines after it
```

```
    adaptline = parstartline + layer
```

```
    parline = fintext[adaptline].split()
```

```
    parline[parcolumn] = "{:.9f}".format(newvalue)
```

```
    #we assume the floats are printed in eighth characters '%8s'
```

```
    parline_new = ['%18s%i' for i in parline]
```

```
    fintext[adaptline] = ".join(parline_new)+'\n'
```

```
    fout.writelines(fintext)
```

```
os.rename(os.path.join(path_to_dir,'Selector_new.in'),os.path.join(path_to_dir,'Selector.in'))
```

```
def replaceTopFlux(path_to_dir, newvalue, parname='rTop', layer=1):
```

```
    try:
```

```
        os.rename(os.path.join(path_to_dir,'Selector.in'),os.path.join(path_to_dir,'Selector_old.in'))
```

```
    except:
```

```
        os.remove(os.path.join(path_to_dir,'Selector_old.in'))
```

```
        os.rename(os.path.join(path_to_dir,'Selector.in'),os.path.join(path_to_dir,'Selector_old.in'))
```

```
    with open(os.path.join(path_to_dir,'Selector_old.in'),'r') as fin:
```

```
        with open(os.path.join(path_to_dir,'Selector_new.in'),'w') as fout:
```

```
            fintext = fin.readlines()
```

```

#Get line with par headers assuming rTop is always a parameter
#using the parameter is not possible, since eg 'n' would give errors
parstartline = fintext.index([x for x in fintext if 'rTop' in x][0])
#Get index (column) of the parameter
parcolumn = fintext[parstartline].split().index(parname)

```

```

#adapting the lines after it
adaptline = parstartline + layer
parline = fintext[adaptline].split()
parline[parcolumn] = "{:.5f}".format(newvalue)
#we assume the floats are printed in eighth characters '%8s'
parline_new = ['%18s%i' for i in parline]
fintext[adaptline] = ".join(parline_new)+'\n'
fout.writelines(fintext)

```

```

os.rename(os.path.join(path_to_dir,'Selector_new.in'),os.path.join(path_to_dir,'Selector.in'))

```

```

def replaceInputIrr(path_to_dir, newvalue, parname='Irrig_rate'):
    layer = 1;
    newvalue = np.around(newvalue,decimals = 3)
    newvalue = newvalue.tolist()
    try:
        os.rename(os.path.join(path_to_dir,'Selector.in'),os.path.join(path_to_dir,'Selector_old.in'))
    except:
        os.remove(os.path.join(path_to_dir,'Selector_old.in'))
        os.rename(os.path.join(path_to_dir,'Selector.in'),os.path.join(path_to_dir,'Selector_old.in'))
    with open(os.path.join(path_to_dir,'Selector_old.in'),'r') as fin:
        with open(os.path.join(path_to_dir,'Selector_new.in'),'w') as fout:
            fintext = fin.readlines()
            #Get line with par headers assuming Irrig_rate is always a parameter
            #using the parameter is not possible, since eg 'n' would give errors
            parstartline = fintext.index([x for x in fintext if 'Irrig_rate' in x][0])
            #Get index (column) of the parameter
            parcolumn = fintext[parstartline].split().index(parname)

            #adapting the lines after it
            adaptline = parstartline + layer
            parline = fintext[adaptline].split()
            parline[parcolumn] = "{:.3f}".format(float(newvalue))
            #we assume the floats are printed in eighth characters '%8s'

```

```

parline_new = ['%18s'%i for i in parline]
fintext[adaptline] = ".join(parline_new)+'\n'
fout.writelines(fintext)

```

```

os.rename(os.path.join(path_to_dir,'Selector_new.in'),os.path.join(path_to_dir,'Selector.in'))

```

```

def deleteIrrig(path_to_dir, newvalue, parname='Irrig', layer=1):

```

```

    try:

```

```

        os.rename(os.path.join(path_to_dir,'Selector.in'),os.path.join(path_to_dir,'Selector_old.in'))

```

```

    except:

```

```

        os.remove(os.path.join(path_to_dir,'Selector_old.in'))

```

```

        os.rename(os.path.join(path_to_dir,'Selector.in'),os.path.join(path_to_dir,'Selector_old.in'))

```

```

    with open(os.path.join(path_to_dir,'Selector_old.in'),'r') as fin:

```

```

        with open(os.path.join(path_to_dir,'Selector_new.in'),'w') as fout:

```

```

            fintext = fin.readlines()

```

```

            #Get line with par headers assuming Irrig is always a parameter

```

```

            #using the parameter is not possible, since eg 'n' would give errors

```

```

            parstartline = fintext.index([x for x in fintext if 'Irrig' in x][0])

```

```

            #Get index (column) of the parameter

```

```

            parcolumn = fintext[parstartline].split().index(parname)

```

```

            #adapting the lines after it

```

```

            adaptline = parstartline + layer

```

```

            parline = fintext[adaptline].split()

```

```

            parline[parcolumn] = newvalue

```

```

            #we assume the floats are printed in eighth characters '%8s'

```

```

            parline_new = ['%7s'%i for i in parline]

```

```

            fintext[adaptline] = ".join(parline_new)+'\n'

```

```

        fout.writelines(fintext)

```

```

os.rename(os.path.join(path_to_dir,'Selector_new.in'),os.path.join(path_to_dir,'Selector.in'))

```

```

def updateProfile(from_path, to_path):

```

```

    """

```

The Hydrus files profile.dat governs the soil profile inputs and nod_inf.out holds the outcome of the soil profile after a Hydrus run.

Parameters


```

-----
from_path:
    Directory with the Hydrus-input and output files where one wants to retrieve soil profile
data.
to_path:
    Directory with the Hydrus-input and output files where one wants to supply soil profile
data.
'''

fout = to_path + str("\profile.dat")
arrout = np.genfromtxt(fout, skip_header = 5, skip_footer = 2)
dfout = pd.DataFrame(arrout, columns=['n','x','h','Mat','Lay','Beta','Axz','Bxz','Dxz'])
dfout = dfout.set_index('n')

fsm = from_path + str("\nod_inf.out")
f = open(fsm,'r')
ft = f.readlines()
st = ft.index([x for x in reversed(ft) if 'Time' in x][0])
fl = [x.split() for x in ft[st+6:-1]]
f1 = [np.asarray(x) for x in fl]
f.close()
dfsm = pd.DataFrame(f1,
columns=['Node','Depth','Head','Moisture','K','C','Flux','Sink','Kappa','v/KsTop','Temp'])
dfsm = dfsm.set_index('Node')
dfout['h'] = dfsm['Head']

try:
    os.rename(os.path.join(to_path,'profile.dat'),os.path.join(to_path,'profile_old.dat'))
except:
    os.remove(os.path.join(to_path,'profile_old.dat'))
    os.rename(os.path.join(to_path,'profile.dat'),os.path.join(to_path,'profile_old.dat'))
with open(os.path.join(to_path,'profile_old.dat'),'r') as fin:
    with open(os.path.join(to_path,'profile_new.dat'),'w') as fout:
        ftext = fin.readlines()
        start = ftext.index([x for x in ftext if 'Beta' in x][0])
        colb = ftext[start].split().index('Beta')-3
        colsm = ftext[start].split().index('Beta')-6

        for layer in range(1,102):
            adaptline = start + layer

```

```

    line = ftext[adaptline].split()
    line[colb] = dfout.get_value(layer,'Beta')
    line[colsm] = dfout.get_value(layer,'h')
    line_new = [%18s%i for i in line]
    ftext[adaptline] = ".join(line_new)+'\n'
    layer +=1
fout.writelines(ftext)

```

```

os.rename(os.path.join(to_path,'profile_new.dat'),os.path.join(to_path,'profile.dat'))

```

```

def movefiles(from_path, to_path):

```

```

    """

```

The Hydrus input files govern the inputs. movefiles takes one set of data and moves it to another folder.

Parameters

```

-----

```

from_path:

Directory with the Hydrus-input and output files where one wants to retrieve data.

to_path:

Directory with the Hydrus-input and output files where one wants to supply data.

```

    """

```

```

shutil.copy(os.path.join(from_path,'selector.in'),os.path.join(to_path,'selector.in'))
shutil.copy(os.path.join(from_path,'profile.dat'),os.path.join(to_path,'profile.dat'))
shutil.copy(os.path.join(from_path,'hydrus1d.dat'),os.path.join(to_path,'hydrus1d.dat'))
shutil.copy(os.path.join(from_path,'ATMOSPHERE.in'),os.path.join(to_path,'ATMOSPHERE.in'))

```

```

def replaceAtm(path_to_dir, newvalues, parname):

```

```

    """

```

The Hydrus files Atmosph.in governs the atmospheric inputs.

Parameters

```

-----

```

from_path:

Directory with the Hydrus-input and output files where one wants to retrieve atmospheric data.

newvalues:

An array of values of the atmospheric parameter that one wants to change.

parname:

A string that provides the parameter one wants to change.

'''

```
newvalues = np.reshape(newvalues, (len(newvalues),1))
```

```
newvalues = np.around(newvalues,decimals = 3)
```

```
newvalues = newvalues.tolist()
```

```
try:
```

```
os.rename(os.path.join(path_to_dir,'ATMOSPHERE.in'),os.path.join(path_to_dir,'ATMOSPHERE_old.in'
```

```
))
```

```
except:
```

```
os.remove(os.path.join(path_to_dir,'ATMOSPHERE_old.in'))
```

```
os.rename(os.path.join(path_to_dir,'ATMOSPHERE.in'),os.path.join(path_to_dir,'ATMOSPHERE_old.in'
```

```
))
```

```
with open(os.path.join(path_to_dir,'ATMOSPHERE_old.in'),'r') as fin:
```

```
with open(os.path.join(path_to_dir,'ATMOSPHERE_new.in'),'w') as fout:
```

```
ftext = fin.readlines()
```

```
start = ftext.index([x for x in ftext if 'tAtm' in x][0])
```

```
parcolumn = ftext[start].split().index(parname)
```

```
for layer in range(1,len(newvalues)):
```

```
adapline = start + layer
```

```
negs = ftext[adapline].replace('-', '-')
```

```
line = negs.split()
```

```
line[parcolumn] = "{:.3f}".format(float(newvalues[layer][0]))
```

```
line_new = ["%18s%i" for i in line]
```

```
ftext[adapline] = ".join(line_new)+"\n"
```

```
layer +=1
```

```
fout.writelines(ftext)
```

```
os.rename(os.path.join(path_to_dir,'ATMOSPHERE_new.in'),os.path.join(path_to_dir,'ATMOSPHERE.i
```

```
n'))
```

```

#-----
# RUNNING HYDRUS
#-----
def runHydrus(guessed_runtime, path_to_dir, install_dir:
    """
    Run the Hydrus model from within Python

    Parameters
    -----
    guessed_runtime:
        runtime of the model, in seconds (take some seconds more)
    path_to_dir:
        path to the working directory with input/output of Hydrus
    install_dir:
        path to the installation directory of the Hydrus software

    """
    didrun = True
    cdtorun=os.path.join(install_dir,'H1D_CALC.EXE')+ ' '+path_to_dir
    proc = subprocess.Popen(cdtorun)

    time.sleep(guessed_runtime) #time nothing is happening to let model run
    proc.terminate()

    if os.path.isfile(os.path.join(path_to_dir,'error.msg')) == True:
        with open(os.path.join(path_to_dir,'error.msg'),'r') as fin:
            fintext = fin.readlines()
            print(fintext)
            didrun = False
    try:
        os.remove(os.path.join(path_to_dir,'error.msg'))
    except OSError:
        pass

    return didrun

```

```

#-----
# RETRIEVING DATA FOR ANALYSIS
#-----

def findIrrIC(path_to_dir):
    """
    Find irrigation applied by the triggered irrigation in Hydrus in an array.

    Parameters
    -----
    path_to_dir:
        path to the working directory with input/output of Hydrus

    """

    with open(os.path.join(path_to_dir,'Irrig.out'),'r') as fin:
        fintext = fin.readlines()

        amt = list()
        line = 'Irrigation amount'
        for x in fintext:
            if line in x:
                x = x.replace('Irrigation amount:',")
                x = float(x)
                amt.append(x)
        amount = np.asarray(amt)

        t = list()
        line = 'Time when'
        for x in fintext:
            if line in x:
                x = x.replace('Time when irrigation is triggered:',")
                x = float(x)
                t.append(x)
        time = np.asarray(t)
        irr = pd.DataFrame({'irrigation':amount}, index = time)
        test = np.zeros(math.ceil(max(time)))
        timet = np.zeros(math.ceil(max(time)))
        for x in range(0,len(test)):

```

```
test[x] = sum(irr.irrigation[x:x+1])
timet[x] = x
matpath = 'C:\\User\\ MATLAB\\HYDRUS\\'
now = datetime.datetime.now()
t = now.strftime("%m-%d-%Y-%H-%M-%S")
string = 'triggirr_day_'+t+'.mat'
matname = os.path.join(matpath,string)
scipy.io.savemat(matname, dict(time = timet, irrigation = test))
return irr
```

Appendix 2: Code from MATLAB

Estimating Soil Moisture (estimsm):

```
function sm = estimsm(q,rain, etc, eto, initialpath)
% Set path to find HYDRUS files
    exp = 'TriggeredIrr_oneday';
    install_dir = 'C:\Program Files (x86)\PC-Progress\Hydrus-1D 4.xx\';
    path_to_dir=strcat('C:\User \HydrusResult\', exp);
% Replace atmospheric in/out files for HYDRUS
    py.definitions.replaceAtm_oneday(path_to_dir,q+rain,'Prec');
    py.definitions.replaceAtm_oneday(path_to_dir, etc,'rRoot');
    py.definitions.replaceAtm_oneday(path_to_dir, eto,'rSoil');

    try
        py.definitions.runHydrus(0.2, path_to_dir, install_dir); % Run HYDRUS
        py.definitions.updateProfile(initialpath, path_to_dir); % Update HYDRUS soil profile
for the next step

        cd('C:\User\MATLAB\HYDRUS');
        Obs = ReadObsNodeData(path_to_dir); % Read data from soil moisture sensor
(observation node in the soil profile)
        sm = Obs(1).theta;
        sm = sm(end);
    catch
        cd('C:\User\ MATLAB\HYDRUS');
        sm = -1;
    end
```

Read Observation Node Data (adapted from code written for HYDRUS and MATLAB by Anjali Jain Figueroa, 2016):

```
function Odata = ReadObsNodeData(desiredpath)
% Set Path
filename = 'OBS_NODE.OUT';
file = fullfile(desiredpath,filename);
fileID = fopen(file,'r');
finfo = dir(file);
% Header
Head_num = 7;
```

```

HeaderText = textscan(fileID,'%s',Head_num,'delimiter','\n');
HeaderLines{1,1} = HeaderText{1};

% Row of labels
Blanks = textscan(fileID,'%s',3,'delimiter','\n');
labels = textscan(fileID,'%s',1,'delimiter','\n');
lbls =strsplit(char(labels{1}(1,1)));
lbls=regexprep(lbls,'/','_');

% Read all Data
while (~feof(fileID))
    InputText = textscan(fileID,'%s','delimiter','\n'); % Read lines
end
fclose(fileID);
% Format Data
for i = 1: size(InputText{1})-1
    line=strsplit(char(InputText{1}(i,:)));
    for j=1:size(lbls,2)-1
        O_data(i,j)=line(j);
    end
end
O_data=str2double(O_data);

% Store in structure
node = 1;
count = 0;
N=size(lbls,2);
totalnodes = N/4; % number of nodes must be specified
for i = 1:N-1
    count = count+1;
    Odata(node).(char(lbls(i)))= O_data(:,i);
    if count == 4;
        node=node+1;
        count = 1;
    end
end
end

```


Minimize Bang-Bang Irrigation:

```
% Set starting parameters for HYDRUS
load('ET_RAIN_00-15.mat','rain', 'croptans', 'evap');
rain = rain';
rain = rain(1:150);
etc = croptans.crop';
eto = evap.evap';
cd('C:\User\Desktop\Code-Python\hydrus_wrapper-master');
exp = 'TriggeredIrr';
path_to_dir=strcat('C:\Users\Desktop\HydrusResult\', exp);
py.definitions.replaceAtm(path_to_dir, rain,'Prec');
py.definitions.replaceAtm(path_to_dir, etc,'rRoot');
py.definitions.replaceAtm(path_to_dir, eto,'rSoil');
cd('C:\User\MATLAB\HYDRUS');

% Set starting parameters for the optimization [irrigation amount, irrigation duration]
x0 = [5 0.5];
lb = [4 0.0069]; %10 minutes, 0.01 cm
ub = [10 1];

% Run optimization with fmincon
[x,fval,exitflag,output] = fmincon(@objfun,x0,[],[],[],[],lb,ub);
savedata = char(strcat('fmincon_505.mat'));
save(savedata,'x', 'fval', 'exitflag', 'output', 'x0');
cd('C:\User\MATLAB\HYDRUS');

Objective Function (objfun):

function f = objfun(x)
    rate = x(1);
    duration = x(2);

    cd('C:\User\Desktop\Code-Python\hydrus_wrapper-master');
    exp = 'TriggeredIrr';
    path_to_dir=strcat('C:\User\Desktop\HydrusResult\', exp);
    install_dir = 'C:\Program Files (x86)\PC-Progress\Hydrus-1D 4.xx';

% Replace input values in HYDRUS files with values from optimizer
py.definitions.replaceInputIrr(path_to_dir, rate, 'Irrig_rate');
py.definitions.replaceInputIrr(path_to_dir, duration, 'Duration');
```

```

try
    py.definitions.runHydrus(0.5, path_to_dir, install_dir);
    cd('C:\User\MATLAB\HYDRUS');
    Obs = ReadObsNodeData(path_to_dir);
    [Obs.time, idx] = unique(Obs.time);
    theta = interp1(Obs.time, Obs.theta(idx), 1:150); % Find values at daily time steps for
each of the 150 day season

catch
    cd('C:\User\MATLAB\HYDRUS');
    theta = ones(1,150)*0.01; % Provide a low value if HYDRUS does not initialize
end

f = theta - ones(1,150)*0.2; % Find e(t) at daily time steps for each of the 150 day season
f = f(f<0); % Find negative error values
f = -sum(f); % Change the sign of the total of the negative error values so the problem
becomes a minimization

```

Find optimal PID parameters:

```

% Set up starting parameters for the optimization [a b c] = [ $\alpha$   $\beta$   $\gamma$ ]
x0 = [4 4 8];
lb = [0 0 0];
ub = [15 15 15];

```

% Run optimization with fmincon

```

options = optimoptions(@fmincon,'StepTolerance', 1e-6);
[x,fval] = fmincon(@objfun_pid,x0,[],[],[],[],lb,ub,...
    [],options);

```

Objective Function to find Optimal PID parameters:

```

function f = objfun_pid(x)
% Parameters for PID optimization
a = x(1);
b = x(2);
c = x(3);

```

```

% Load nominal irrigation from bang-bang optimization and set up HYDRUS with reference and
crop evapotranspiration
load('min505.mat', 'M');

```

```

Qnom2 = [M M];

load('min505.mat','croptans','evap','rain');
rain = rain';
etc = croptans.crop';
eto = evap.evap';
etc = [etc etc];
eto = [eto eto];

% Set up HYDRUS to day-by-day for 150 days
cd('C:\User\Desktop\Code-Python\hydrus_wrapper-master');
exp = 'oneday';
expic = 'oneday_IC';
path_to_dir=strcat('C:\User\Desktop\HydrusResult\', exp);
initial=strcat('C:\User\Desktop\HydrusResult\', expic);

% Run PID
cd('C:\User\MATLAB\HYDRUS');
thetanom = 0.2;
Qi = 0;
Q(1) = 2.5; Q(2) = 2.25;
theta(1) = estimsm_413(Q(1),rain(1), etc(1), eto(1),initial);
theta(2) = 0.2576;
theta(3) = 0.2469;
theta(4) = 0.2390;
theta(5) = 0.2407;
itheta = sum(theta-thetanom);

for i = 3:149
    Q(i) = Qnom2(i) - a*(theta(i-1) - thetanom)- b*(itheta) - c*(theta(i-1)-theta(i-2));
    theta(i) = estimsm(Q(i),rain(i), etc(i), eto(i),path_to_dir);
    if i >= 6
        itheta = sum(theta(i-4:i)-ones(1,5)*thetanom);
    else
        itheta = sum(theta-thetanom);
    end
    if Q(i) <= 0
        Q(i) = 0;
    end
    Qi = Qi+Q(i);
end
% Return to optimizer to minimize total irrigation
f = Qi;

```

