Release Notes for AquaCrop

Version 7.3
April 2025

Developed for the

Food and Agriculture Organisation of the United Nations

by

Kunz RP
Centre for Water Resources Research
School of Agricultural, Earth and Environmental Sciences
University of KwaZulu-Natal, Pietermaritzburg, South Africa

April 2025

SUMMARY

Background

A four-year project funded by the Water Research Commission (WRC) began in April 2023 to develop an online database of simulated data for 13 underutilised indigenous crops. Simulations will be performed using a crop simulation model originally developed in 2009 by the Food and Agriculture Organisation (FAO) called AquaCrop. For each crop, the model will be run for five planting dates, each with two plant densities representing small- and large-scale production. The model is run for 49 consecutive seasons for each of the 5838 Altitude Zones (AZs) across southern Africa. Due to the very large number of seasonal model runs required (i.e. 13 crops *x* 5 planting dates *x* 2 plant densities *x* 49 seasons *x* 5838 AZs), an important objective of the research project is to "develop and implement a fast and efficient method to run FAO's AquaCrop model".

The stand-alone (SA) version of the AquaCrop model will be used, which when compared to the GUI version, runs faster and also allows for model runs to be automated (i.e. batched). In August 2022, the SA program was re-written in the Fortran programming language and released as v7.0, which runs much faster than previous versions (e.g. v6.0) written in Pascal. Version 7.1 was released in August 2023 but could not be used to create the crop database because the model produces no output when the crop simulation period is reduced. This major issue was fixed in v7.2, which was released by FAO in August 2024. However, the latest version still crashes unexpectedly due to various "bugs". Therefore, work began in mid-March 2025 to modify the open source code of v7.2 to (i) prevent the model from stopping unexpectedly (i.e. crashing) due to various bugs, and (ii) reduce model run time by minimising disk I/O. This report represents a detailed description of the (i) rigorous testing undertaken for AquaCrop SA (cf. Section 2), and the (ii) various changes made to the model's code to create v7.3 (cf. Section 3), which runs considerably faster than v7.2 (cf. Section 4).

Throughout this report, words highlighted in **bold** refer to (i) files required as input, or (ii) files created by AquaCrop, as well as (iii) folder names where these files are stored/created (i.e. files external to the model). All names of variables, subroutines and functions are highlighted in *italics* (i.e. internal to the model). Severe model bugs are highlighted in red. Of particular importance are numerous recommendations highlighted in yellow that FAO should consider to further improve AquaCrop. A summary of these recommendations is presented next.

Important bug fixes to consider (cf. Section 3.1)

Versions 7.1 and 7.2 underwent rigorous testing that identified various bugs that were fixed as follows:

- in the ***season.out** file (**OUTP** folder),
  - remove an unknown variable to fix the alignment of monthly with seasonal simulations;
  - change Tot(x) to Tot(xxx) to improve alignment for more than 10 (and 100) seasons;
- stop the model from crashing if, in the multiple project (**.PRM**) file (**LIST** folder),
  - folder names do not end in a backslash (\);

    - file names are set to (none/external) instead of (None)/(External);
- stop the model from
    - running if the plant density is zero (instead of simulating zero transpiration biomass production and yield formation);
    - hanging if run in a cold climate (due to very slow accumulation of growing degree-days);
    - crashing if the soil description in the **.SOL** file (**DATA** folder) is a single word, e.g. sand, silt or clay;
- ensure that the determination of the crop cycle defaults to calendar day mode to prevent inconsistent model behaviour.

Performance improvements (cf. Section 3.2)

In order to reduce model run time (i.e. increase model performance), disk I/O was minimised as follows:

- in the **DATA** folder, read/access the following files only once (and not each season):
    - each climate (**.ETo**, **.PLU** and **.TNX**) file, then store the entire data record in a global array;
    - **.SOL** file;
    - **.CO2** file (store each value in a global array);
- in the **SIMUL** folder, stop the creation of the
    - **EToData.SIM**, **RainData.SIM** and **TempData.SIM** files and instead, store the seasonal datasets in three global arrays;
    - ***Reference.TnX**, **TnxReference365Days.SIM**, **TCropReference.SIM** and **TCrop.SIM** files and **again,** store the data in three global arrays;
    - **DEFAULT.CRO** and **DEFAULT.SOL** files multiple times (i.e. once per season);
- modifying various subroutines and functions to utilise data stored in these global arrays, rather than accessing the physical files during the simulation of each season.

Prevent the model from crashing/hanging (cf. Section 3.3)

After significantly improving the model's performance by reducing disk I/O, the next step involved preventing AquaCrop SA from:

- running indefinitely, i.e. stuck in a permanent loop due to a problem with the temperature data;
- crashing due to
    - division-by-zero (two cases);
    - array-out-of-bound errors (four cases).

Other minor code changes (cf. Section 3.4)

Other minor code changes were implemented such as:

- commenting out code (e.g. subroutines and functions) no longer used;
- in the **OUTP** folder, the
    - file **AllDone.OUT** is no longer created (reduces disk I/O);
    - phrase "All done" is written to **ListProjectsLoaded.OUT** file;
    - heading "Projects handled:" is written to the **ListProjectsLoaded.OUT** file.

- updating the version number and release date to 7.3 and April 2025, respectively.

Testing of v7.3 (cf. Section 4.1)
The modified version (v7.3) of AquaCrop produces identical results to the official version (v7.2) for

- the test case provided by FAO (multiple cuttings of alfalfa across three years in Ottawa, Canada);
- 28 Altitude Zones across South Africa.

It is important to note that v7.3 has NOT been tested using 10-day or monthly climate input data.

Reduction in model run time (cf. Section 4.2)
Minimising disk I/O resulted in a significant reduction in model run time, considering SA v7.3 (written in Fortran) runs 259 and 160 times faster than the GUI v7.2 and SA v6.0 programs respectively (both of which are written in Pascal). Similarly, v7.3 is ~20x and 25x faster than v7.1 and v7.2 respectively (which are both coded in Fortran). When compared to v7.2, v7.3's model run time is reduced by 96%.

Other recommendations to FAO
It is hoped that FAO will incorporate most of the recommended changes (above-mentioned) into a future release of the model. A summary of other recommendations are as follows:

- To further reduce disk I/O, similar changes should be made to the code to ensure the **.CRO**, **.IRR**, **.MAN**, **.GWT**, **.SW0**, **.OFF** and **.OBS** files are read in once and only again if the current file name does not match the previous file name.
- The **DEFAULT.CRO** file included in the test case on the Github website does not match the file created by v7.2.
- The GUI version of AquaCrop should be modified to allow the user to restrict the season length (calculated using thermal time in GDD units)
    - to a user-specified value (e.g. 365 or 730 days), which is particularly important when running the model in cold climates not ideally suited to crop production;
    - when the minimum air temperature drops below a certain threshold, i.e. immediately stop growth for crops that are frost sensitive.
- The GUI and SA programs should be altered to increase the size of arrays storing actual and reference temperature data (e.g. *Tm**CropReferenceRun* and *Tm**TnxReference365DaysRun*) from 365 to a minimum of 366 days (to account for a leap year), but preferably 1096 days. This is important to prevent numerous array-out-of-bound errors.
- The user should be made aware that for v7.2 onwards, it is important to provide climate files with as much data record as possible, so that the calculation of long-term means of monthly temperature are representative of the location.
- Since all of the temporary climate files are no longer created in the **SIMUL** folder, variables, subroutines and folders that refer to the old files should be deleted.
- Regarding the **TCrop.SIM** file (now a global array called *TCropdotSIM*):

- it does not contain data for the entire cropping period stipulated in the **.PRM** file, and
- is not used since the variable *ReferenceClimate* is always set to true.

Some of the code changes suggested in this report represent temporary fixes so that v7.3 produces the same results as v7.2. However, they should be fixed properly in a future release, e.g. two division-by-zero errors, which result from dominator values that should not be zero (such as soil water content in the root zone).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AZ	Altitude Zone
CCL	Crop Cycle Length
CWP	Crop Water Productivity
CWRR	Centre for Water Resources Research
D	Difference
DAP	days after planting
FAO	Food and Agriculture Organisation (of the United Nations)
FC	Field Capacity
FO	Frost Occurrence
GUI	Graphic User Interface
SA	Stand-Alone (AquaCrop executable)
SL	Season Length
TT	Thermal Time
UKZN	University of KwaZulu-Natal
WRC	Water Research Commission
WSL	Windows Subsystem for Linux

LIST OF SYMBOLS

B	above-ground biomass production	dry t ha^{-1}
CCL	crop cycle length	GDDs
CWP	crop water productivity	dry kg ha^{-1}
E	soil water evaporation	mm
ET_O	reference crop evapotranspiration	mm
GDD	growing degree-day	°C day
HI	harvest index	%
SL_{TT}	season length calculated using thermal time	DAP
SL_{FO}	season length reduced by frost occurrence	DAP
Tr	crop transpiration	Mm
WPet	crop water productivity	dry kg m^{-3}
Y	crop yield	dry t ha^{-1}

1 INTRODUCTION

1.1 Background

In April 2023, the Centre for Water Resources Research (CWRR) based at the University of KwaZulu-Natal (UKZN) in Pietermaritzburg (South Africa) started a four-year project funded by the Water Research Commission (WRC). The main aim of this project is to develop an online database of simulated data for underutilised indigenous crops. A simple web-based utility will provide, inter alia, farmers and policy makers with easy (and free) access to the crop database. The use of the crop database should help remove barriers currently preventing the adoption of underutilised crops as a source of income and nutritious food for both small- and large-scale farmers in South Africa. Access to the database should also help promote and support the cultivation of underutilised crops, thus improving agricultural diversification in the country. This research project harnesses the considerable investment made by the WRC over the past decade in field work and modelling of underutilised crops. This investment has resulted in (i) the calibration of a crop simulation model for various underutilised crops, and (ii) the ability to run the model at a national scale for 5838 Altitude Zones (AZs), where each zone is considered a relatively homogenous response zone.

Previous research funded by the WRC focused on a crop simulation model originally developed in 2009 by the Food and Agriculture Organisation (FAO) called AquaCrop (Hsiao et al., 2009; Raes et al., 2009; Steduto et al., 2009), which was initially applied to maize (Heng et al., 2009; Hsiao et al., 2009) and quinoa (Geerts et al., 2009). In South Africa, AquaCrop has been used to simulate crop water use and yield of numerous crops. In 2015 for example, AquaCrop simulations for each AZ were used to develop maps of rainfed yield and water productivity for several crops (Kunz et al., 2015). In 2017, AquaCrop was run with climate projections from four CMIP3 GCMs to assess the impact of climate change on three underutilised crops (bambara nut, sorghum and taro) and three commercial crops (canola, soybean sugarcane). This work was published in a handbook for the agricultural sector on adaptation to climate change (Kunz and Schulze, 2017; Mabhaudhi et al., 2017ab; Schulze, 2017). Kunz et al. (2020) ran AquaCrop for soybean and sorghum for all AZs, each with two planting dates and two plant densities. Runs were also undertaken for both rainfed and irrigated growing conditions. The latter model runs were used to develop a unique set of monthly crop coefficients for each AZ, which were then used as input for the ACRU hydrological model to assess the impact of rainfed crop production on downstream water availability. AquaCrop was run with downscaled and bias-corrected future climate projections from six CMIP5 GCMs to assess the impact of climate change on four underutilised crops across three 30-year time periods, i.e. 1961-1990, 2015-2044 and 2070-2099 (Kunz and Mabhaudhi, 2023). Kunz et al. (2024) ran AquaCrop for two underutilised crops (sweet potato and taro), each with two planting dates and two plant densities for both rainfed and irrigated conditions. Similar runs have also been completed for another four crops (maize, potato, soybean and groundnut). This should explain why AquaCrop is the most widely used crop simulation model in South Africa (Kephe et al., 2021).

1.2 Methodology

The above-mentioned national-scale AquaCrop model runs were undertaken using versions 4 and 6 of the stand-alone (SA) program, which were released by the FAO in June 2012 and March 2017 respectively. When compared to the model program with a graphic user interface (GUI), the SA program not only runs faster but also allows for model runs to be automated (i.e. batched). Since 2014, approximately 10000 lines of code have been developed in Unix and Fortran to fully automate the national-scale model runs. For example, a program called GenPrm was developed to create individual project (.PRO) files (in the **LIST** folder) for each cropping season. In addition, specialised software

monitored for any pop-up window displaying an error (e.g. division-by-zero) message, which was then automatically acknowledged and closed, thus allowing the model to automatically run for the next season.

For this project, the AquaCrop model will again be used to simulate data for selected underutilised crops, together with the climate and soil databases for all AZs across southern Africa (Eswatini, Lesotho and South Africa). The climate database contains 50 years of daily climate data (rainfall, temperature and reference evapotranspiration), thus allowing for 49 consecutive seasonal simulations (1950/51 to 1998/99) in each AZ. For two soil horizons, the soil database provides saturated hydraulic conductivity as well as soil water retention values at saturation, field capacity and wilting point (Kunz et al., 2024). Statistics generated from monthly and seasonal output for 14 AquaCrop variables will be stored in the crop database for 13 underutilised crops. The model will be run for five planting dates from the 1st of September to the 1st of January, each with two plant densities representing small- and large-scale production of each crop. Hence, a very large number of seasonal model runs are required (i.e. 13 crops *x* 5 planting dates *x* 2 plant densities *x* 49 seasons *x* 5838 AZs), which again requires the use of SA model.

In August 2022, the SA program was re-written in the Fortran programming language and released as v7.0. The development of this open source code version was funded by the European Union's Horizon 2020 research and innovation programme (grant agreement no. 773903). Version 7.1 was released in August 2023 but could not be used to create the crop database because the model produces no output when the crop simulation period is reduced. This major issue was fixed in v7.2, which was released by FAO in August 2024. However, the latest version still crashes unexpectedly due to various "bugs".

Owing to the large number of model runs required to develop the crop database, an important objective of the current research project is to "develop and implement a fast and efficient method to run FAO's AquaCrop model". Version 7 written in Fortran runs much faster than previous versions written in Pascal. Work began in mid-March 2025 to modify the source code of v7.2 to (i) prevent the model from stopping unexpectedly (i.e. crashing) due to various errors, and (ii) reduce model run time by minimising disk I/O. This report represents a detailed description of the (i) rigorous testing undertaken for AquaCrop SA (cf. Section 2), and the (ii) various changes made to the model's code to create v7.3 (cf. Section 3), which not only produces identical results to v7.2 (cf. Section 4.1), but also runs considerably faster (cf. Section 4.2).

Throughout this report, words highlighted in **bold** refer to (i) files required as input, or (ii) files created by AquaCrop, as well as (iii) folder names where these files are stored/created (i.e. files external to the model). All names of variables, subroutines and functions are highlighted in *italics* (i.e. internal to the model). Severe model bugs are highlighted in red. Of particular importance are numerous recommendations highlighted in yellow that FAO should consider to improve AquaCrop.

2 TESTING OF AQUACROP v7

2.1 Background

The stand-alone (SA) program of AquaCrop has been rigorously tested in South Africa by comparing output against that produced using the GUI program. This has been done for v6.0, as well as for versions 7.1 and 7.2. Version 6.0 of the SA program produces identical results to the GUI (v6.0) program, which is expected since both programs are written in the same programming language, namely Pascal. Thus, output from the SA v6.0 program was used as a benchmark for testing SA v7.1 (and v7.2), which are both written in the Fortran programming language.

Tests were conducted using a locally calibrated parameter file for soybean, where the crop is planted on the 1st of November in each season at a density of 317460 plants ha^{-1}. Climate and soil data were obtained from the Altitude Zones climate and soil databases (Kunz et al., 2024). The climate database provides 50 years (1950-1999) of daily rainfall, temperature (maximum and minimum) and reference evapotranspiration (ET_O) data for each Altitude Zone (AZ). The altitudinal variation across each zone is minimal, which implies minimal variation in climate and soil. Hence, the AZs are ideal for crop simulation modelling, since each zone can be considered a relatively homogenous response zone. The soil database provides saturated hydraulic conductivity as well as soil water retention parameters at saturation (SAT), field capacity (FC) and wilting point (WP), for two soil horizons up to a total maximum depth of 1.2 m. The location of 28 AZs used for model testing is shown in Figure 2-1, which were selected based on problems experienced in the past when running versions 4 and 6.

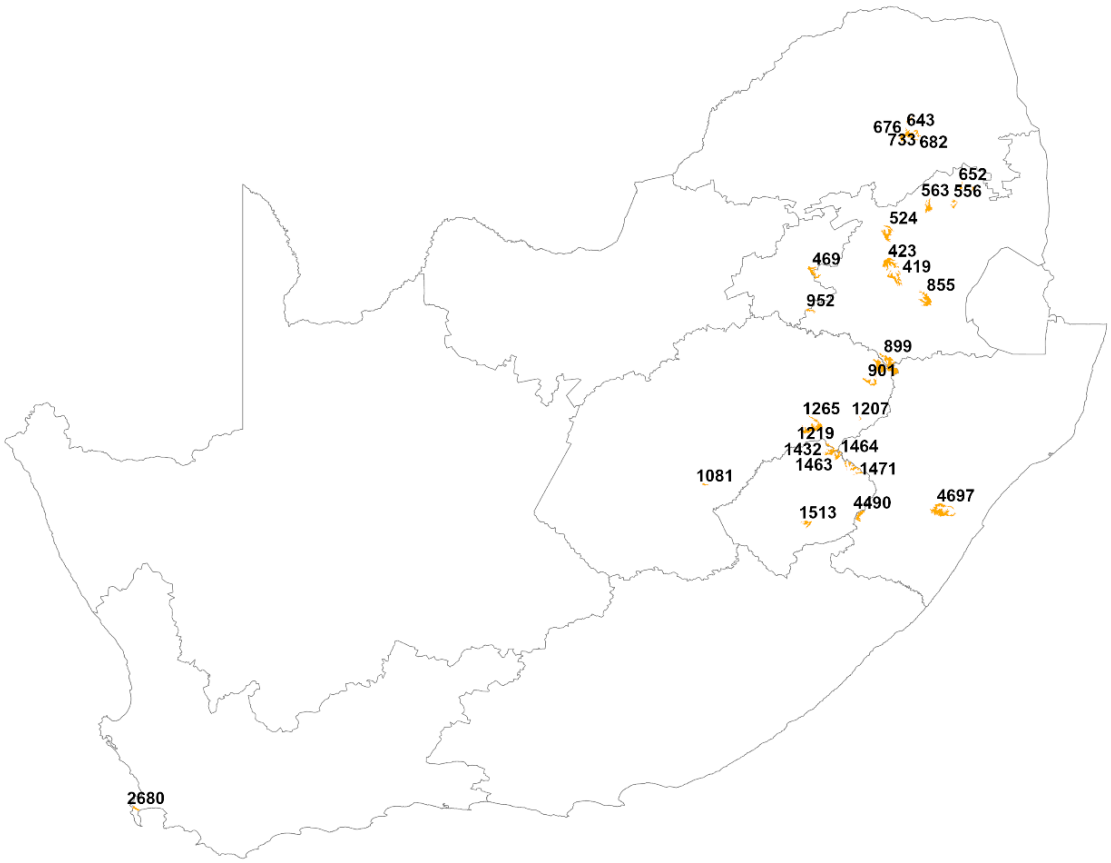

Figure 2-1 Location of the 28 Altitude Zones used for testing the AquaCrop model

Some of the AZs surrounding Lesotho (e.g. AZ 1432) are at high altitude where air temperatures are too cold for economically viable crop cultivation. However, it is important to test the model in such zones where season lengths extend well beyond 365 days. In AZ 1432 for example, GDDs accumulate very slowly, resulting in very long season lengths that range from 675-773 (average 731) days. Hence, the GUI program only simulates 18 (of 49) consecutive seasons. The model is run in such areas to determine their suitability for crop cultivation. Model output can be used to develop land suitability maps, as demonstrated by Lake (2024).

2.2 Testing of v7.1

Version 7.1 of the AquaCrop SA program was released in August 2023. In May 2024, the model underwent testing using AZs 4696 (higher altitude; cooler) and 4697 (lower altitude; warmer). Season length (SL) is defined as the number of days **after planting** to when the crop reaches physiological maturity, which is best determined using thermal time in growing degree-day (GDD) units. The time to reach physiological maturity in GDDs is a crop-specific input parameter (e.g. 2025 GDDs for soybean). The crop cycle length (CCL), which represents the number of days **after emergence** to physiological maturity, is always shorter than the SL calculated using thermal time (TT), i.e. by 5-8 days as shown in Table 2-1.

Table 2-1 Difference (D) in season length (SL) and crop cycle length (CCL), expressed in days after planting (DAP) and calculated using thermal time (TT) for Altitude Zones 4696 and 4697

Season		4696 (DAP)			4697 (DAP)		
		SL_{TT}	CCL	D	SL_{TT}	CCL	D
19	1968/69	127	122	5	122	116	6
20	1969/70	136	130	6	128	121	7
27	1976/77	128	120	8	120	113	7
30	1979/80	141	134	7	130	125	5
42	1991/92	130	123	7	122	116	6
45	1994/95	130	123	7	123	117	6
49	1998/99	130	122	8	123	116	7

AquaCrop SA v6.0 and 7.1 were run for the 19th season (1968/69) in AZ 4697. The crop reached physiological maturity at 122 days after planting (DAP) on the 3rd of March 1969 (cf. Table 2-1), after the accumulation of 2025 GDDs. The model was then run for a SL ranging from 1 to 132 DAP. The additional 10 days showed that, as expected, transpiration does not increase after the crop has reached physiological maturity. However, when the season length was artificially reduced (i.e. 1-121 DAP), AquaCrop SA v7.1 unexpectedly crashed on line 5419 in global.f90 due to an end-of-file error, and thus no output was simulated. The error occurred in function *SeasonalSumOfKcPot*() when reading T_{min} and T_{max} data from the file called **TCrop.SIM** (created by the model in the **SIMUL** folder). This is because this file only contained temperature data from the first to the last day of simulation, which if reduced, is insufficient to calculate GDDs to maturity.

As shown in Figure 2-2, no transpiration (Tr in mm) was simulated by v7.1 of the standalone (SA) program when the SL was shorter than 121 DAP. This was different to AquaCrop SA v6.0, which simulated transpiration after the crop had emerged from 6 DAP onwards.

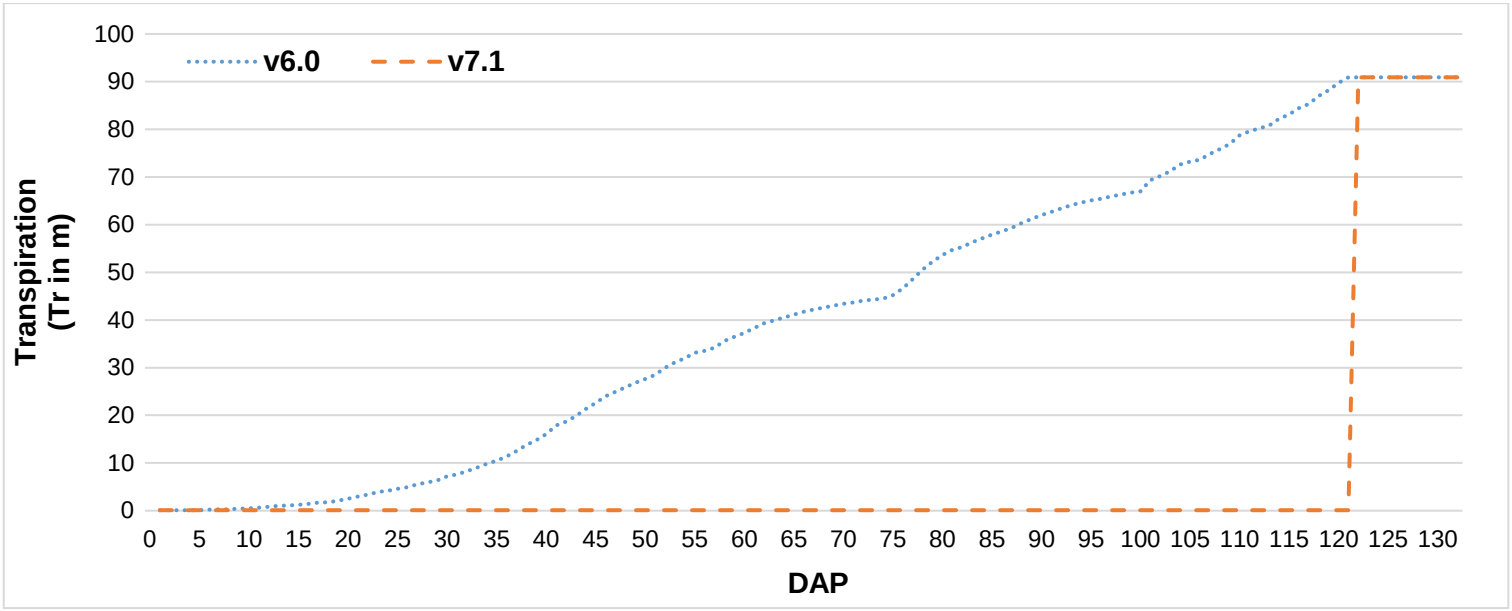

Figure 2-2 AquaCrop SA v7.1 vs v6.0 simulations of transpiration from 1 to 132 days after planting (DAP) in the 19th season (1968/69) for Altitude Zone 4697

Since above-ground biomass production (B) is calculated from accumulated Tr, v7.1 simulated no B before the crop reached physiological maturity (Figure 2-3). Similarly, since yield (Y) is calculated from B via the HI, no Y was simulated by v7.1 of the model before 122 DAP. In comparison, AquaCrop SA v6.0 simulated Y from 72 DAP onwards, reaching a maximum value of 0.682 dry t ha^{-1} when the crop matured (Figure 2-4).

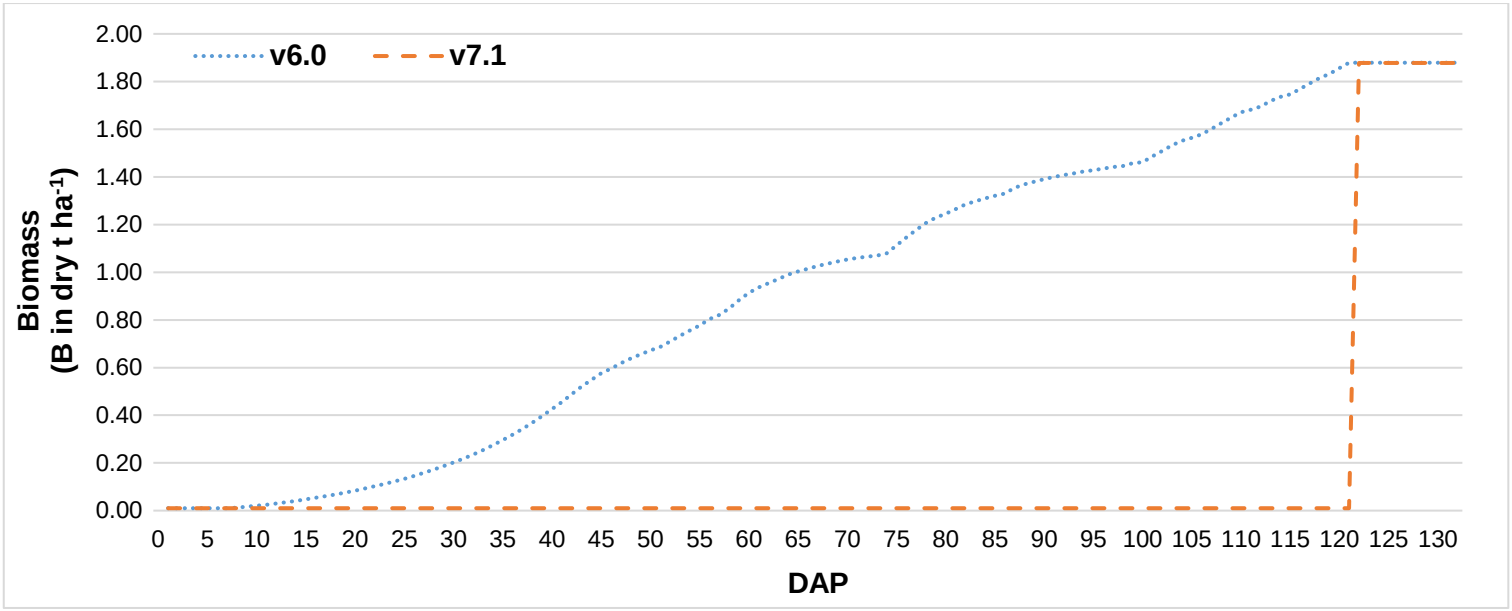

Figure 2-3 AquaCrop SA v7.1 vs v6.0 simulations of biomas production from 1 to 132 days after planting (DAP) in the 19th season (1968/69) for Altitude Zone 4697

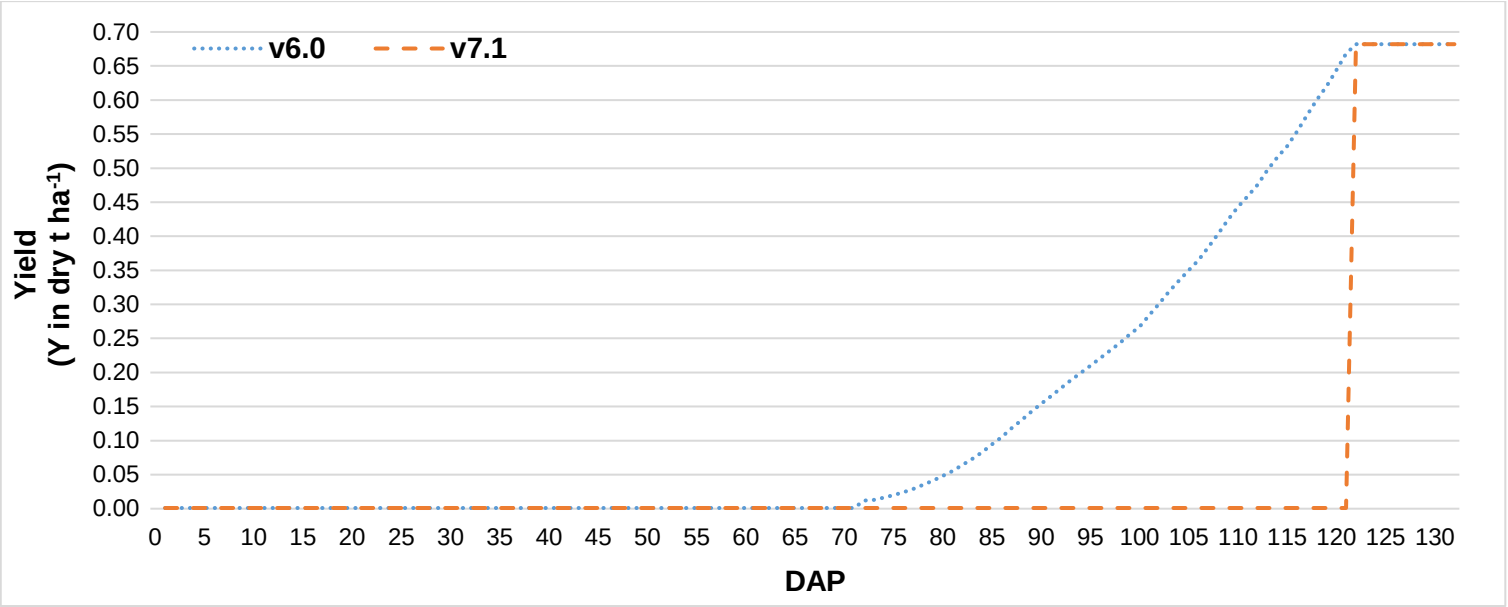

Figure 2-4 AquaCrop SA v7.1 vs v6.0 simulations of yield formation from 1 to 132 days after planting (DAP) in the 19th season (1968/69) for Altitude Zone 4697

The above-mentioned problem represented a severe bug with v7.1 of the SA program, and thus this version was deemed unsuitable for the large scale model runs required for various research projects in South Africa. The decision was made to wait for FAO to release v7.2, which was then tested again.

2.3 Testing of v7.2

The latest version of AquaCrop (v7.2) was released in August 2024. According to the release notes for this version (FAO, 2024), the model produces slightly different results to the previous version (v7.1) because of the following:

a) The decline in biomass with increasing soil fertility stress now uses long-term averages of daily maximum and minimum temperature (calculated from the entire temperature data file), and not data for the simulation period.
b) Similarly, the reduction in maximum canopy cover (CCx) with increasing soil salinity stress for unstressed conditions is now based on long-term averages of daily maximum and minimum temperature (not data for the simulation period).

FAO (2024) tested the above changes by simulating winter wheat production in Tunis (Tunisia) using 22 years of data from 1979-2001. The yield differences between version 7.1 and 7.2 ranged from -0.6 to 1.8%, with the average below 0.50%.

2.3.1 Alfalfa (Ottawa, Canada)

FAO provided a test case representing alfalfa grown in Ottawa (Canada), with output created by v7.2. Multiple cuttings are harvested over a three year period (2014-2016). Differences in yield and biomass ranged from -0.9 to 1.6%, whilst differences in crop water productivity ranged from -1.2 to 1.9% (Table 2-2).

Table 2-2 Differences in AquaCrop output between v7.1 and v7.2 of multiple cuttings of alfalfa grown over a three year period in Ottawa, Canada

Variable	v7.1	v7.2	% difference
Biomass (dry t ha^{-1})	9.320	9.172	1.6
	11.842	11.947	-0.9
	12.466	12.579	-0.9
Yield (dry t ha^{-1})	9.162	9.014	1.6
	11.842	11.947	-0.9
	12.466	12.579	-0.9
Crop water productivity (dry kg m^{-3})	2.19	2.15	1.9
	2.44	2.47	-1.2
	2.65	2.67	-0.7

Source (v7.2) data: https://github.com/KUL-RSDA/AquaCrop/tree/main/testcase

However, the **DEFAULT.CRO** file produced by versions 7.1 and 7.2 in the **SIMUL** folder were different to file included in the test case for one parameter, namely the "Crop performance under elevated atmospheric CO2 concentration (%)", where the value changed from 50 to 100%. Hence, FAO should update this file on the Github website where the test case can be downloaded from.

As expected, v7.2 compiled by FAO produced identical results to the test case (except for the **DEFAULT.CRO** file). The source code for AquaCrop v7.2 was downloaded from the Github website and compiled using Intel's Fortran compiler for Windows (v2021.6.0). The locally compiled version produced identical output to the test case. This version was then modified to fix numerous bugs initially identified when testing v7.1, as explained in Section 3.1.

2.3.2 Soybean (AZ 1432, South Africa)

As noted in Section 2.1, it is important to test AquaCrop in areas that are too cold for economically viable crop production, as it highlights numerous problems with the program. AquaCrop GUI v7.2 was run for AZ no. 1432 to simulate soybean growth and yield in the 1st season (1950/51). The crop reaches physiological maturity on the 9th of January 1953 (800 DAP). The model simulated a utilisable (seed) yield of 1.605 dry t ha^{-1} from 9.623 dry t ha^{-1} of biomass production (HI of 17%), as shown in Table 2-3. These values are unrealistic due to the very long SL being economically unviable.

The model runs were than repeated for all 18 consecutive seasons. The SL ranged from 700-800 (average of 760) DAP, whereas the CCL, which excludes the time to emergence, ranged from 675-773 (average of 732) DAP. The GUI program outputs the following variables as integers, whereas the SA program outputs values to one decimal: Rain, ETo, GDD, Infilt, Runoff, Drain, E, Tr, TrW. For these variables, slight differences (< 0.5 mm) occurred, whereas for all other output variables, both programs produced identical results. AquaCrop GUI took ~111 seconds to complete, whereas the SA program completed the same task in 4.273 s.

The need to limit the maximum season length

Seasonal yield simulations can incorrectly indicate that the zone is suited to crop (e.g. soybean) production, yet SLs are unrealistically long. To prevent AquaCrop SA from simulating unrealistic SLs, the GenPrm program was designed to limit the season length to a set value. For most annual crops, limiting the season to a maximum of 364 DAP is a

reasonable assumption to prevent unrealistic yield simulations from crop cycles that span years in length. This limit also makes sense considering the reference temperature files created by v7.2 in the **SIMUL** folder contain data for 365 days. Limiting the maximum SL to a user-specified value also improves model performance.

Model runs were repeated for SLs shortened to 730 and 365 days. Simulations produced by the GUI program were then compared to output from the SA program for v7.2, as shown in Table 2-3. The minor difference in output between the GUI and SA programs (highlighted in italics in the table below) is mostly due to the way in which Pascal and Fortran handle the rounding of floating point calculations. It is therefore recommended that the GUI version of AquaCrop is modified by FAO to allow the user to restrict the SL to a user-specified value.

Table 2-3 Impact on AquaCrop simulations when the season length is reduced for soybean planted in the 1st season (01 November 1950) in Altitude Zone 1432

AquaCrop v7.2 model output	**Season length (DAP)**					
	800		**730**		**365**	
	GUI	**SA**	**GUI**	**SA**	**GUI**	**SA**
Biomass production (dry t ha^{-1})	9.623	9.623	8.781	*8.778*	1.724	*1.723*
Utilisable (seed) yield (dry t ha^{-1})	1.605	1.605	1.464	1.464	0.000	0.000
Harvest index (%)	16.7	16.7	16.7	16.7	0.0	0.0

Limiting the season length to 365 days will also reduce the likelihood of the model generating an array-out-of-bound error. For sugarcane grown in South Africa, the season length can range from 12 to 24 months. Hence, FAO are encouraged to increase the size of arrays storing actual and reference temperature data (e.g. *Tm**CropReferenceRun*, *Tm**TnxReference365DaysRun*, etc.) from 365 to a minimum of 366 days (to account for a leap year), but preferably 1096 days.

Version 7.1 of the SA program crashed when the season length was reduced (cf. Section 2.2). This serious bug was fixed by FAO in v7.2, where the function *SeasonalSumOfKcPot*() in global.f90 (line 5457) was modified to rather use daily temperature data from a new file called **TCropReference.SIM** (also created in the **SIMUL** folder) and not **TCrop.SIM**. This new file is created in subroutine *DailyTnxReferenceFileCoveringCropPeriod*() in preparefertilitysalinity.f90. It contains daily temperature data for 365 days from the day of planting onwards (i.e. 1st of November to 31st of October). This prevents the end-of-file error when **TCrop.SIM** is used, which only has temperature data for the simulation period. The data in **TCropReference.SIM** is identical to that stored in **TnxReference365Days.SIM**, which has daily values from 1st of January to 31st of December. The daily data is generated from long-term monthly means calculated from the entire temperature (**.TNX**) file, which are stored in a file called ***Reference.Tnx** in the **SIMUL** folder. Hence, sufficient temperature data exists to calculate the crop maturity date, even if the season length is reduced to 365 days (for example).

2.3.3 Soybean (AZ 4697, South Africa)

A maximum season length of 365 days is, however, unrealistic for annual crops that are frost sensitive (e.g. soybean), considering the plant is unlikely to survive frost events that make occur from late autumn into winter. For this reason, the GenPrm program has the ability to end the season if the minimum air temperature (measured at 2 m above ground level) drops below a certain threshold.

Consideration of frost occurrence

Weather data measured at the University of KwaZulu-Natal in Pietermaritzburg (South Africa) was used by Lake (2024) to develop a relationship between surface temperature (at 0 m) and air temperature (measured at 2 meters above the ground surface). As shown in Figure 2-5, ground temperature is typically below 0°C when the air temperature is 5.4°C or less. Hence, GenPrm can reduce the season length if the minimum air temperature reaches 5.4°C or less before the crop reaches physiological maturity. This threshold temperature is a user-specified input for the GenPrm program.

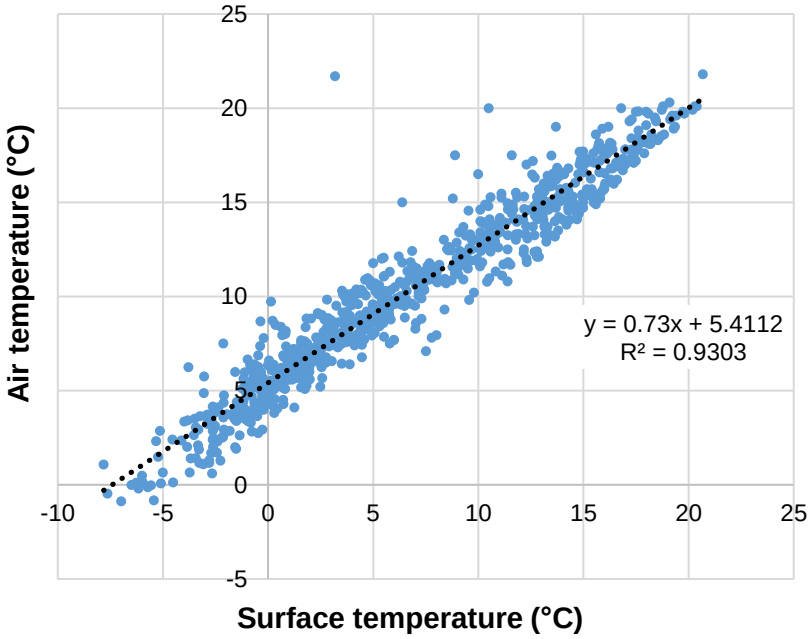

Figure 2-5 Relationship between minimum air and surface temperature based on data measured by an automatic weather station in Pietermaritzburg from from April to August between 2020 to 2024 (Lake, 2024)

The city of Pietermaritzburg is located in AZ 4697, and thus tests were conducted for zone 4696 (higher altitude; cooler) to zone 4698 (lower altitude; warmer). Table 2-4 highlights the impact of frost occurrence (FO) on season length (SL_{FO}), which was drastically reduced across seven seasons in zone 4696, due to the possibility of frost occurring between 9 and 28 DAP, i.e. after 01 November. Similarly, the SL was reduced in three seasons (19, 27 and 30) in zone 4697 (Table 2-4). Minimum temperatures in zone 4698 are above 5.4°C, and thus SL is not affected by FO.

Table 2-4 Likely reduction in season length (SL) caused by frost occurrence (FO) for Altitude Zones 4696 and 4697

Season	4696 (DAP)			4697 (DAP)		
	SL_{TT}	SL_{FO}	D	SL_{TT}	SL_{FO}	D
19	127	9	118	122	12	110
20	136	11	125	128		
27	128	25	103	120	25	95
30	141	9	132	130	9	121
42	130	21	109	122		
45	130	28	102	123		
49	130	21	109	123		

AquaCrop SA v7.2 was further tested using climate data for the 19th season in AZ 4697. As expected, the model no longer crashed when the SL was artificially reduced. The model produced almost identical output to v6.0 (Figure 2-6), except for minor differences (e.g. $0.0 \leq Tr_{diff} \leq 0.1$ mm; $-0.001 \leq B_{diff} \leq 0.001$ dry t ha^{-1}; $Y_{diff} = 0.000$ dry t ha^{-1}) related to the way in which the Pascal (v6.0) and Fortran (v7.2) programming languages round floating point numbers.

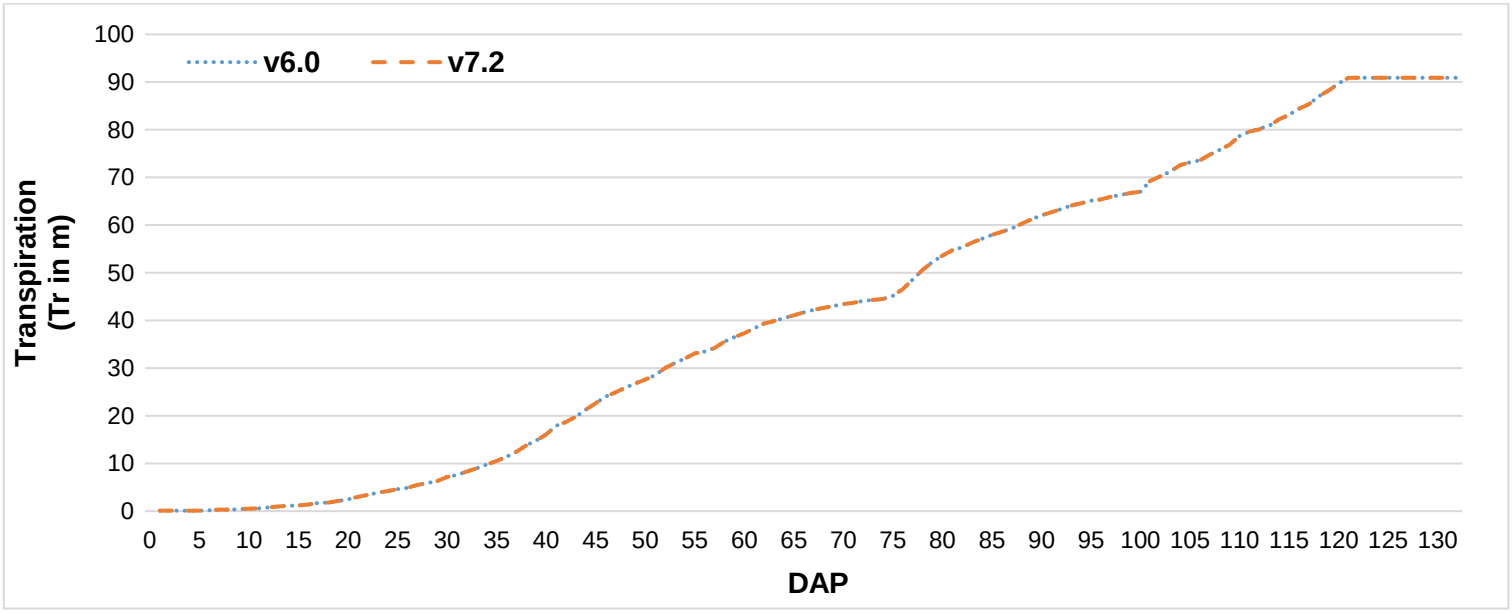

Figure 2-6 AquaCrop SA v7.2 vs v6.0 simulations of transpiration from 1 to 132 days after planting (DAP) in the 19th season (1968/69) for Altitude Zone 4697

The reason why v7.2 produced almost identical results to v6.0 is both SA models were run with 365 days of climate data from 1st of November 1968 to 31st of October 1969. Hence, the long-term averages of monthly temperature were generated from **only one year** of data (**not 50 years**). The SA programs were run with one year of data to improve their performance (i.e. reduce model run time) since both models run progressively slower for each simulated season. For the 19th season, the model reads the climate file from the beginning (1st of January 1950) until it reaches the required planting date (1st of November 1968). For the 49th season, the mode "skips" over the majority of data stored in the temperature (**.TNX**) file. It is therefore recommended that v7.2 is run with the full climate file available for the location, despite the significant impact on model performance. This needs to be made much clearer in the release notes for v7.2 (FAO, 2024).

From Figure 2-6, a SL reduced from 122 to only 12 DAP by the first occurrence of frost (cf. Table 2-4) will result in negligible transpiration, and thus no biomass production or yield formation. It is therefore recommended that the GUI version is modified by FAO to allow the user to restrict the SL due to frost occurrence using a user-specified minimum temperature value.

3 DEVELOPMENT OF AQUACROP v7.3

3.1 Initial bug fixes

During the testing phase involving 28 AZs (cf. Section 2.1), it was important to determine what can cause AquaCrop SA v7.2 to stop working unexpectedly, i.e. crash. This information was then used to modify the model's Fortran code to fix these anomalies (i.e. bugs), which are described next. Each fix can be identified in the modified code by searching for a comment (!) called "RPK fix", followed by a two-digit number from 01 to 10 (e.g. "! RPK fix 01").

1) As shown in the Table 3-1, an unexpected bug was discovered in AquaCrop SA v7 where the monthly output has an extra variable in the ***season.out** file. This unknown variable occurs after WPet and has a value of -9 (i.e. missing/unknown), which causes misalignment of the columnar data when compared to the seasonal output. This unknown variable was easily identified in the code and removed by making the following correction:

subroutine *WriteTheResults*() in *run.f90*

```
! RPK fix 01
write(TempString, '(f9.1, 4i9, 2f9.3)') HI, undef_int, undef_int, undef_int, &
                                  undef_int, BmobPer, BstoPer
```

to

```
write(TempString, '(f9.1, 3i9, 2f9.3)') HI, undef_int, undef_int, undef_int, &
                                  BmobPer, BstoPer
```

Table 3-1 Difference between monthly and seasonal output from AquaCrop version 7.2, where an unknown output variable is highlighted in red

HI %	Y(dry) ton/ha	Y(fre) ton/ha	WPet kg/m3	Unknown variable	Bin ton/ha	Bout ton/ha
0.0	-9	-9	-9	-9	0.000	0.000
0.0	-9	-9	-9	-9	0.000	0.000
6.5	-9	-9	-9	-9	0.000	0.000
18.9	-9	-9	-9	-9	0.000	0.000
38.1	-9	-9	-9	-9	0.000	0.000
38.1	2.156	-9.900	0.43		0.000	0.000

2) The model crashes (with error code 2) if path names in the **.PRM** file do not end in a backslash (\). This bug was fixed by adding the following code:

subroutine *read_project_file*() in *project_input.f90*

```
integer :: len ! RPK fix 02
! the forward slash works for both Windows and Linux operating systems with Intel's Fortran compiler
(ifort)
len=len_trim(buffer) ! RPK fix 02
if (buffer(len:len) /= ')' .and. (buffer(len:len) /= '\' .or. buffer(len:len) /= '/')) buffer(len+1:len+1)
= '/'
```

The fix was added after each line that reads in a directory/folder name stored in the variable called buffer (14 times in total).

3) AquaCrop SA v6 crashes if the plant density is set to 0 planta ha^{-1}, whereas v7 does not and outputs zero transpiration, biomass and yield. This bug was fixed by adding the following code:

function *GetCrop_PlantingDens* in *global.f90*

```
! RPK fix 03
if (PlantingDens == 0) then
    write(*,*) 'ERROR: planting density cannot be 0 plants/ha'
    stop 1
   endif
```

4) If the “Determination of crop cycle” parameter is changed from 0 (growing degree-day mode) or 1 (calendar day mode) to say -9 (for example), the program crashes for certain (**but not all**) seasons. This inconsistency was first noticed in v7.1, but was fixed in version 7.2 by the developers (i.e. FAO) as follows:

function *GetCrop_PlantingDens* in *global.f90*

```
! RPK fix 04 - the crop cycle mode always defaults to calendar day mode
! crop%ModeCycle = ModeCycle = 1 (CalendarDays mode) if not zero
read(fhandle, *) XX
if (XX == 0) then
    call SetCrop_ModeCycle(ModeCycle_GDDays)
else
    call SetCrop_ModeCycle(ModeCycle_CalendarDays)
end if
```

5) If missing file names in the **.PRM** file are set to (**n**one) instead of (**N**one) for headings 1, 2, 3 and 10, the model “crashes” with error code 0 and the following files are created in the **OUTP** folder:
- **AllDone.OUT**;
- **ListProjectsLoaded.OUT**, which indicates the project (**.PRM**) file was NOT loaded; and
- **ProjectPRMseason.OUT** that contains four lines of header information, as well as monthly and seasonal output.

This is not expected since AquaCrop SA v7.2 produces output in a file called **ProjectPRMseason.OUT** instead of **AltZne_4697_PRMseason.OUT** yet indicates the **.PRM** file was not loaded (i.e. run).

If missing file names in the **.PRM** file are set to (**n**one) instead of (**N**one) for headings 4, 5, 7, 8 and 9, the model crashes with error code 2 and the following files are created in the **OUTP** folder:
- **AllDone.OUT** is not created
- **ListProjectsLoaded.OUT** that indicates the project (**.PRM**) file was NOT loaded; and
- **ProjectPRMseason.OUT** that contains four lines of header information only (i.e. no monthly or seasonal output).

This bug was also fixed by checking if the file name is (None) and changing it to (none). In addition, if the file name is (External), it is changed to (external) as follows:

subroutine *read_project_file*() in *project_input.f90*

```
! RPK fix 05
! added after each line that read in a file name OR directory name stored in the buffer variable (14 times
in total)
if (trim(buffer) == '(None)') buffer = '(none)'
if (trim(buffer) == '(External)') buffer = '(external)'

replace (None) with (none) in
global.f90                    ! 25 replacements
initialsettings.f90           ! 12 replacements
preparefertilitysalinity.f90  ! 03 replacements
run.f90                       ! 24 replacements
```

```
tempprocessing.f90              ! 27 replacements

replace (External) with (external) in
global.f90                      ! 05 replacements
preparefertilitysalinity.f90    ! 04 replacements
run.f90                         ! 06 replacements
tempprocessing.f90              ! 22 replacements
```

6) The model crashes unexpectedly in certain seasons and no ***season.OUT** file is created and the **ListProjectsLoaded.OUT** file is empty. These crashes are most likely due to division-by-zero errors. Some division-by-zero errors were corrected, which is discussed later in more detail (see **Section 3.3**).

7) From previous experience, AquaCrop SA must be tested in very cold AZs (e.g. zone 1432) where there are insufficient heat units (growing degree-days or GDDs) for the crop to reach physiological maturity in a reasonable time frame (e.g. 365 days). When AquaCrop SA v7 was run for zone 1432, it ran indefinitely (i.e. stuck in an infinite loop), which is a serious bug. This bug was addressed as explained in **Section 3.3**.

8) In the ***season.OUT** file, the season number was changed to a 3-digit number, i.e. Tot(1) was changed to Tot(001). This makes it easier to post-process the output files since the season number is consistent in length.

```
subroutine WriteTheResults() in run.f90
! RPK fix 08
write(TempString, '(i0)') ANumber
to
write(TempString, '(i3.3)') ANumber
```

9) AquaCrop SA v7.1 and v7.2 unexpectedly crashed in zone 0676 and 2680 when reading in the **.SOL** file. This caused the model to produce no simulations for all 49 seasons, which is a serious bug.

```
Altitude Zone 0676
        3.0                 : AquaCrop Version (August 2012)
       46                   : CN (Curve Number)
       05                   : Readily evaporable water from top layer (mm)
        2                   : number of soil horizons
        0.20                : Depth (m) of restrictive soil layer inhibiting root zone expansion
  Thickness  Sat   FC    WP     Ksat        description
  ---(m)-   ----(vol %)-----  (mm/day)  --------------------------------
    0.09    43.9  18.7  11.0   897.8    Sandy Loam
    0.11    42.5  20.9   9.9   630.1    Silt

Altitude Zone 2680
        3.0                 : AquaCrop Version (August 2012)
       46                   : CN (Curve Number)
       05                   : Readily evaporable water from top layer (mm)
        2                   : number of soil horizons
        0.26                : Depth (m) of restrictive soil layer inhibiting root zone expansion
  Thickness  Sat   FC    WP     Ksat        description
  ---(m)-   ----(vol %)-----  (mm/day)  --------------------------------
    0.12    44.9  15.7   7.9  1439.2    Sandy Loam
    0.14    45.1  19.4   8.7  1046.0    Silt
```

The model also crashed with the following soil file packaged with v7.2.

```
Davis soil, ClayL layers FC inc. to 33% WP dec. to 13.8%, ready E=8 mm,3 m depth
        3.0                 : AquaCrop Version (June 2012)
       60                   : CN (Curve Number)
        8                   : Readily evaporable water from top layer (mm)
        3                   : number of soil layers
       -9.00                : Depth (m) of restrictive soil layer inhibiting root zone expansion - None
  Thickness  Sat   FC    WP     Ksat        description
  ---(m)-   ----(vol %)-----  (mm/day)  --------------------------------
    1.80    51.0  33.0  13.8   100.0     YoloClayLoam
    0.30    35.0  20.0  10.0   150.0     YoloSandyLoam
    0.90    51.0  33.0  13.8   100.0     YoloClayLoam
```

The above three soil files have a common issue where the soil description is a single word (e.g. Silt or YoloClayLoam). Hence, the model would also crash if the soil description is either a Sand or Clay. The serious bug occurs in subroutine *LoadProfile*() in the *global.f90* file, where the variable called "*blank*" was removed from the read statement as follows:

```
subroutine LoadProfile() in global.f90
! RPK fix 09 : ! for version 3 (or 4) soil files, a blank is read in after Ksat which is incorrect
read(fhandle, *) thickness_temp, SAT_temp, FC_temp, &
                 WP_temp, infrate_temp, blank, description_temp
to
read(fhandle, *) thickness_temp, SAT_temp, FC_temp, &
                 WP_temp, infrate_temp, description_temp
```

10) The soil (**.SOL**) file is read 50 times, i.e. initially, then again for each of the 49 seasons. This also occurs for the CO_2 (**.CO2**) file. To prevent this from happening, the following code was placed around the call statements to specific subroutines:

```
! RPK fix 10 ! only read the file once
if (NrRun == 1) then
    call SetCO2File() or
    call SetProfFle() or
    call LoadProfile()
end if
```

This represents a temporary fix and a better solution would involve re-reading the **.SOL** file if the current file name was different to the previous name.

3.2 Performance improvements

3.2.1 Reading in climate data into global arrays

From experience, AquaCrop (both the GUI and SA programs) runs progressively slower for each consecutive season that is simulated. This issue is particularly noticeable when running the model with 139 years of future climate projections spanning 1961 to 2099. This was confirmed for v6.0 and v7.1 when a single **.PRM** file (for all seasons) was converted into individual **.PRO** files (one per season). The model run time for each season was obtained using the Unix time command, which showed a large difference between the first and last season. This indicated that the model reads the climate file from the beginning for each season. As a workaround for v6.0, a utility called CliGet was written in Fortran to extract climate data for a particular season from the large climate (**.ETo**, **.PLU** and **.TNX**) files. The model run time for each season

was far more consistent, with slight fluctuations due to season length. A better solution is to read the climate data into an array, which should only be done once, thus preventing unnecessary disk I/O, especially if the climate files are accessed each season.

In v7.2, the daily temperature file is accessed four times in two subroutines as follows:

1) *ReadTemperatureFilefull*() in *tempprocessing.f90* counts the number of lines in the **.TNX** file,
2) *ReadTemperatureFilefull*() then re-reads the temperature record for each season,
3) *CreateTnxReferenceFile*() in *tempprocessing.f90* also reads the temperature record to create long-term monthly means, which are stored in a file called ***Reference.Tnx** in the **SIMUL** folder, and
4) *CreateTnxReferenceFile*() creates two temperature files (each with 365 days of data) in the **SIMUL** folder called **TCropReference.SIM** and **TnxReference365Days.SIM**.

Similarly, subroutine *LoadClim*() in *global.f90* reads the **.ETo** and **.PLU** files, again once for each season. In total, the climate files are accessed a total of 6 times per season, which represents 294 times for 49 seasons. This high disk I/O significantly slows down the program, especially if it is run on a slow hard drive (i.e. 5400 RPM hard drive in a laptop). Hence, additional code changes were made to reduce disk I/O, and thus vastly improve AquaCrop's performance. These code changes, as described in this sub-section, can be found by searching the **.f90* files for "! RPK fix 20". The code changes were made incrementally (a to e below), allowing for the compiled version to be checked against the test case provided by FAO (cf. **Section 2.3.1**), before the next code was implemented.

20) **Step a**: In *tempprocessing.f90*, subroutine *ReadTemperatureFilefull*() was deleted and replaced with subroutine *LoadClim*() in *global.f90*. Some code from *ReadTemperatureFilefull*() were moved to *LoadClim*() to handle the temperature (**.TNX**) file. A new variable called *FileType* was introduced to indicate which climate file is being read in:

```
FileType = 1 for .ETo file
FileType = 2 for .PLU file
FileType = 3 for .TNX file
```

Step b: Subroutine *LoadClim*() is first called from *tempprocessing.f90* to read in the entire temperature record. It counts the number of rows in the **.TNX** file to set the size of two new allocatable arrays called *TminFull* and *TmaxFull*, which store the entire daily temperature record. Each array is indexed by row number, i.e. 1 to the number of lines in the data file. Both the *TminFull* and *TmaxFull* are declared as global arrays in global.f90, so that they can be accessed in any subroutine as follows:

```
use ac_global, only:    TminFull, &  ! RPK fix 20
                        TmaxFull     ! RPK fix 20
```

Step c: As noted previously, subroutine *LoadClim*() is called once per season. To reduce disk I/O, this was changed to only call *LoadClim*() if the climate file name is different to the previous one. Hence, a new global variable called *TemperatureFileFullPrevious* was used to store the previous file name. These changes were repeated for the **.ETo** and **.PLU** files. Hence, global variables *EToFileFullPrevious* and *RainFileFullPrevious* were also created. Allocatable arrays with the full ETo and rainfall datasets are called *EToFull* and *RainFull*, respectively. Both global variables are allocated when the number of rows/lines of temperature data is determined.

Step d: The next step involved changing other subroutines to rather use the global arrays and not access the climate files for each season. For example, subroutine *CreateDailyClimFiles*() in *run.f90* was simplified to use the daily climate data stored in the global arrays called:

```
ETofull     (current file name is ETofileFull;          previous file name is ETofileFullPrevious)
Rainfull    (current file name is RainfileFull;         previous file name is RainfileFullPrevious)
Tminfull    (current file name is TemperaturefileFull;  previous file name is TemperaturefileFullPrevious)
Tmaxfull    (current file name is same for TMinFull)
```

To further reduce disk I/O, it is recommended that similar changes are made to the code to ensure the **.CRO**, **.IRR**, **.MAN**, **.GWT**, **.SW0**, **.OFF** and **.OBS** files are read in once and only again if the current file name does not match the previous file name.

Step e: In subroutine *InitializeGlobalStrings*() in *global.f90*, the following new strings were initialised:

```
- SetRainFileFullPrevious('')
- ETofileFullPrevious('')
- SetTemperatureFileFullPrevious('')
```

Summary: To re-cap, subroutine *LoadClim*() is run first from *tempprocessing.f90* to read the temperature data file. The entire data record is stored in a global array and used in other subroutines, e.g. *CreateTnxReferenceFile*(). *LoadClim*() is then run from *run.f90* to read in the entire daily ETo and rainfall data records. The **.TNX** file is now accessed twice, and the **.ETo** and **.PLU** files are accessed once (i.e. 4 in total), regardless of the number of seasons. This significantly reduced disk I/O, resulting in a substantial reduction in model run time. Initial speed tests on a slow computer showed a reduction in model run time by 62%, which means the modified version of AquaCrop SA now runs 2.6 times faster.

3.2.2 Stop the creation of seasonal climate files (called *Data.SIM)

AquaCrop creates a number of temporary files in the **SIMUL** folder. For example, **EToData.SIM** is created by subroutine *LoadClim*() in *global.f90*, which contains ETo data for each season. The seasonal ETo data is then accessed in subroutine *OpenClimFilesAndGetDataFirstDay*() in *run.f90*. Similarly, **RainData.SIM** and **TempData.SIM** also contain seasonal data, and thus are created multiple times. To further reduce disk I/O, these relatively small files are no longer created and rather, climate data stored in the global arrays is used when necessary. The code changes described in this sub-section can be found by searching for "! RPK fix 21". Again, the code changes were implemented in incremental steps (a to d below), allowing for checking in-between each step.

21) **Step a**: Subroutine *OpenClimFilesAndGetDataFirstDay*() in *run.f90* was modified to use the **Full* global arrays that store the entire climate data record. This subroutine retrieves the first day's data from the ***Data.SIM** files. AquaCrop calculates a number to represent each date, where the 1st of January 1901 is day number 1. The day number prior to the start of the climate record is stored at the beginning of each global array (in position zero). Hence, day number 17897 is stored in array position 0, which represents the 31st of December 1949. ETo data for the *FirstDay* (day number 18202 for 1st of November 1950) is located in array position 305, i.e. *FirstDay – ETofull(0)* = 18202 – 17897 = 305.

Step b: Similarly, subroutine *ReadClimateNextDay*() in *run.f90* is called from subroutine *FileManagement*() and was also modified to use the **Full* global arrays. This subroutine retrieves data from the ***Data.SIM** files

for a particular day number. This is made easier using existing global variables within AquaCrop, such as *DayNri* (current simulation day), *Simulation_FromDayNr* (start of simulation period for a particular season) and *Simulation_ToDayNr* (end of simulation period).

Step c: Subroutine *CreateDailyClimFiles*() in *run.f90* was modified to stop creating each seasonal **EToData.SIM**, **RainData.SIM** and **TempData.SIM** file in the **SIMUL** folder. In addition, subroutine *FinalizeRun2*() was modified since there is no need to close these temporary climate files, since they are no longer created.

Step d: Due to the above changes, the following subroutines and functions in *run.f90* are no longer needed, and thus were commented out:

- subroutine *CloseClimateFiles*()
- subroutine *f*SIM_open*()
- function *f*SIM_read*()
- subroutine *f*SIM_close*()

Similarly, the following local variables are no longer needed:

- integer :: *f*SIM* ! file handle
- integer :: *f*SIM_iostat* ! IO status

where * is ETo, Rain or Temp.

Summary: To re-cap, various subroutines were modified to stop the creation of temporary climate files (***Data.SIM**) in the **SIMUL** folder. Instead, seasonal data is retrieved from the global climate arrays (**Full*). This further reduces disk I/O and resulted in an overall reduction in model run time by 68% (including the changes reported in Section 3.2.1), which means the modified version of AquaCrop SA now runs 3.1 times faster.

3.2.3 Stop the creation of other temporary files in the SIMUL folder

AquaCrop creates other temporary files in the **SIMUL** folder, which contains long-term monthly and 365 days of daily temperature data. To further reduce disk I/O, AquaCrop was modified to stop the creation of these files. Once again, code changes were made in incremental steps (a to I below) to rather use data stored in global arrays. The changes described in this sub-section can be found by searching for "! RPK fix 22".

22) **Step a**: Subroutine *CreateTnxReferenceFile*() in *tempprocessing.f90* reads in the temperature file, then calculates long-term monthly averages and stores them in a filed called ***Reference.Tnx** in the **SIMUL** folder. The subroutine was modified to utilise the *TminFull* and *TmaxFull* global arrays, thus negating the need to access the **.TNX** file again (cf. Section 3.2.1 for more detail).

Step b: Subroutine *CreateTnxReferenceFile*() was also modified to not create ***Reference.Tnx** since the long-term mean monthly temperature values are already stored in two global variables called:

- *TminTnxReference12MonthsRun*(i)
- *TmaxTnxReference12MonthsRun*(i)

where *i* is the month from 01 to 12.

Step c: Subroutine *GetMonthlyTemperatureDataSetFromTnxReferenceFile*() in *tempprocessing.f90* also uses the above two arrays, but *i* represents the previous month. Since ***Reference.Tnx** is no longer created, the following subroutines are no longer needed:

```
- subroutine fTnxReference_open()        ! opens the file
- subroutine fTnxReference_write()       ! creates the file
- subroutine fTnxReference_close()       ! closes the file
- subroutine fTnxReference_erase()       ! deletes the file
```

Calls to the above subroutines from *tempprocessing.f90* and *preparefertilitysalinity.f90* were commented out (i.e. removed) to further reduce disk I/O and improve model performance.

Step d: Since the ***Reference.Tnx** file is no longer created, the name of the file cannot be set in subroutine *SetTnxReferenceFileFull*, nor retrieved by subroutine *GetTnxReferenceFileFull*(). This caused AquaCrop to crash with an error, which occurred in subroutine *DailyTnxReferenceFileCoveringCropPeriod*() in *preparefertilitysalinity.f90*. The AquaCrop code was searched for references to *GetTnxReferenceFileFull*() and the following code was commented out to prevent this error:

```
! if (FileExists(GetTnxReferenceFileFull())) then
! end if
```

A similar issue was found in subroutine *CreateTnxReference365Days*() in *tempprocessing.f90*, and thus the following code was also commented out:

```
! if ((FileExists(GetTnxReferenceFileFull())) .OR. (GetTnxReferenceFile() == '(external)')) then
! end if
```

However, the following in *global.f90* are still required:

```
- subroutine SetTnxReferenceFileFull()
- function GetTnxReferenceFileFull()
- subroutine SetTnxReferenceFile()
- function GetTnxReferenceFile()
```

This needs to be fixed in a future release by searching for *ReferenceFile* and removing (commenting out) any call to the above-mentioned subroutines and functions.

Step e: AquaCrop creates another file called **TnxReference365Days.SIM** in the **SIMUL** folder, which stores daily values generated from the long-term monthly means. **TnxReference365Days.SIM** is created in subroutine *CreateTnxReference365Days*() in *tempprocessing.f90*, which is not needed since the data is already stored in two global arrays called:

```
- TminTnxReference365DaysRun(i)
- TmaxTnxReference365DaysRun(i)
```

Hence, subroutine *CreateTnxReference365Days*() was modified to stop the creation of **TnxReference365Days.SIM**.

Functions already exist to set/get:

```
- all 365 values:     ***Tm**TnxReference365DaysRun()
- one day's value:    ***Tm**TnxReference365DaysRun_i()
```

where *** is either Get/Set and ** is either in/ax.

Step f: AquaCrop creates a similar file called **TCropReference.SIM** in the **SIMUL** folder, except the values are re-arranged to start from the planting date onwards, i.e. from the "onset", which is variable *RefCropDay1*. Since the 1st of November is day 305 (in a non-leap year), this file contains data from **TnxReference365Days.SIM** for days 305 to 365, followed by data from day 1 to 304. **TCropReference.SIM** is created in subroutine *DailyTnxReferenceFileCoveringCropPeriod*() in *preparefertilitysalinity.f90*. Again, this file is not needed because the data is stored in two existing global arrays called:

- *TminCropReferenceRun*(i)
- *TmaxCropReferenceRun*(i)

Hence, subroutine *CreateTnxReference365Days*() was modified to stop the creation of **TCropReference.SIM**.

Subroutine *DailyTnxReferenceFileCoveringCropPeriod*() is called from the following two subroutines in *preparefertilitysalinity.f90*:

- subroutine *ReferenceStressBiomassRelationship*()
- subroutine *ReferenceCCxSaltStressRelationship*()

Step g: The function *SeasonalSumOfKcPot*() in *global.f90* was modified to not use the temporary file called **TCropReference.SIM** (since it is no longer created), but rather to obtain values from the two existing global arrays. Functions already exist to set/get:

- all 365 values: ****Tm**CropReferenceRun*()
- one day's value: ****Tm**CropReferenceRun_i*()

where *** is either Get/Set and ** is either in/ax.

Step h: AquaCrop creates another temporary file in the **SIMUL** folder called **TCrop.SIM**. This file stores temperature data for the cropping period (not the simulation period). When the cropping period is identical to the simulation period (which is most often the case), **TCrop.SIM** is identical to **TempData.SIM**. However, the cropping period can be longer than the simulation period. For AZ 0419, the simulation period in the first season starts on 1st of November 1950 and ends on 8th of April 1951 when the crop reaches physiological maturity (i.e. 159 days in total or 158 DAP). The start and end of the cropping period were set to the 1st of January 1950 (day no. 17898) and the 30th of April 1951 (day no. 18382) respectively, i.e. 485 days in length. However, **TCrop.SIM** only contained data for 229 days from 1950/01/01 to 1950/08/17 (day no. 17898-18126), which was not expected. This issue needs to be addressed in a future release.

Subroutine *TemperatureFileCoveringCropPeriod*() in *tempprocessing.f90* was modified to not create **TCrop.SIM** as a file, but rather to store the daily temperature values in a new global array called *TCropdotSIM*, with the number of days of data is stored in array position zero). Subroutine *BTransferPeriod*() in *tempprocessing.f90* was also modified to use the new global array (*TCropdotSIM*), not **TCrop.SIM** since the file is no longer created:

```
Tndayi = TCropdotSIM(Dayi,1)      ! minimum air temperature on Dayi
Txdayi = TCropdotSIM(Dayi,2)      ! maximum air temperature on Dayi
```

Function *SeasonalSumOfKcPot*() in *global.f90* was also modified to use the new global array (not the **TCrop.SIM** file). Daily minimum and maximum temperature values stored in **TCropReference.SIM** are used if the variable *ReferenceClimate* is set to true, else data from **TCrop.SIM** is used. Hence, the code the code was changed to:

```
if (ReferenceClimate .eqv. .true.) then
    ! use temperature values from TCropReference.SIM file
    Tndayi = GetTminCropReferenceRun_i(j)
    Txdayi = GetTmaxCropReferenceRun_i(j)
else
    ! use temperature values from TCrop.SIM file
    Tndayi = TCropdotSIM(j,1)
    Txdayi = TCropdotSIM(j,2)
end if
```

A similar change was made to function *BNormalized*() in *tempprocessing.f90* to use the new global array (*TCropdotSIM*) if *ReferenceClimate* is set to true.

All **.f90* programs were searched for "*ReferenceClimate*" to make sure all code was modified accordingly. However, *ReferenceClimate* is always set to .true. when the function *Bnormalized*() in *tempprocessing.f90* is called from subroutines:
- *StressBiomassRelationshipForTnxReference*(), and
- *CCxSaltStressRelationshipForTnxReference*()

in *preparefertilitysalinity.f90*. Thus, AquaCrop will always use the 365 days of temperature data stored in the *GetTm**CropReferenceRun* global arrays (i.e. former **TCropReference.SIM** file) and not data stored in the new *TCropdotSIM* global array (i.e. former **TCrop.SIM** file). If *ReferenceClimate* was set to false, AquaCrop would stop working since the *TCropdotSIM* array would contain insufficient temperature data and the model would crash. This issue is linked to the major bug in v7.1 that was described in Section 2.2. In a future release, the code should therefore be simplified to always use the *Tm**CropReferenceRun* data.

Step i: AquaCrop also creates two files with default crop and soil parameters in the **SIMUL** folder called **DEFAULT.CRO** and **DEFAULT.SOL**, respectively. These two files are created each season, yet they do not change since they only contain default values. The following changes were made to subroutine *ResetDefaultCrop*() and *ResetDefaultSoil*() in *defaultcropsoil.f90* to only create these files once (if they do not exist), thus reducing disk I/O:

```
if (.not.FileExists(GetProfFilefull())) call call SaveProfile(GetProfFilefull())
if (.not.FileExists(GetCropFilefull())) call call SaveCrop(GetCropFilefull())
```

If user wants the two default files re-created, they should be deleted before AquaCrop SA is run.

Summary: To re-cap, various subroutines were modified to stop the creation of temporary files storing temperature and default data in the **SIMUL** folder. Instead, seasonal data is retrieved from either existing or new global arrays and default crop and soil files are only created once (if they do not exist). This further reduced disk I/O and resulted in an overall reduction in model run time by 80% (including the improvements described in Sections 3.2.1 and 3.2.2), which means the modified version of AquaCrop SA now runs 5.1 times.

3.2.4 Accessing the .CO2 file

All **.f90* files were searched for the string "rc) ! Description" to check if climate-related files are read anywhere else in the code, like in subroutine *CreateTnxReferenceFile*(), which is now fixed (refer to Step a and Step b in Section 3.2.3). However, in function *CO2ForSimulationPeriod*() in *global.90*, the **.CO2** file is

read in once per season. The code was modified in incremental steps (a to e below) to reduce disk I/O. The changes described in this sub-section can be found by searching for "! RPK fix 23".

23) **Step a**: A new subroutine called *LoadCO2*() was created in *global.f90* to read in values from the **.CO2** file and store them in a new global array called *CO2Full*. The **.CO2** file is only read once unless the file name changes. As for the **.ETo**, **.TNX** and **.PLU** files, a new global variable called *CO2FileFullPrevious* was used to store the previous file name. A new function and subroutine were also added to set and get (retrieve) the name of the previous **.CO2** file:

```
- function   GetCO2FileFullPrevious() result(str)
- subroutine SetCO2FileFullPrevious(str)
```

Step b: The new subroutine *LoadCO2*() is called from *tempprocessing.f90* and was modified to interpolate CO_2 values for missing years (e.g. 1903-1904, 1906-1911, 1916-1923 etc.). The interpolation code was copied from subroutine *CO2ForSimulationPeriod*(), then simplified to improve performance.

Step c: Function *CO2ForSimulationPeriod*() in *global.f90* was also simplified to use the already interpolated CO_2 values stored in the new *CO2Full* global array.

Step d: The **.CO2** file is read again in function *CO2ForTnxReferenceYear*() in *preparefertilitysalinity.f90*. This function was also simplified to use the CO_2 values stored in *CO2Full*, thus further reducing disk I/O.

Step e: In subroutine *InitializeGlobalStrings*() in *global.f90*, the following new variable (previous **.CO2** file name) was initialised:

```
- SetCO2FileFullPrevious('')
```

Summary: AquaCrop is now far more efficient at reading in atmospheric CO_2 data from the **.CO2** file and interpolating values in missing years, as this is only done once (not each season). Tests were conducted to ensure the modified version produced the same interpolated CO_2 values and the same output as the original SA program.

3.3 Preventing AquaCrop from crashing

After significantly improving the model's performance by reducing disk I/O, the next step involved preventing AquaCrop SA from (i) running indefinitely (stuck in a permanent loop due to no/insufficient temperature data; see step a below) and crashing mainly due to (ii) division-by-zero (steps b and c), and (iii) array-out-of-bound errors (steps d to g). These problems were identified by debugging the code using Microsoft's Visual Studio Code Editor. As noted in Section 2.1, the model was run for particular AZs where, in the past, AquaCrop SA v6.0 (and v7.1) crashed multiple times. The code changes described in this sub-section (steps a to g) can be found by searching for "! RPK fix 24".

24) **Step a**: Function *SumCalendarDaysReferenceTnx*() in *preparefertilitysalinity.f90* has a do while loop that can run indefinitely if the variable *RemainingGDDays* never becomes 0 (or negative). A new loop counter called *j* was added to exit the do while loop if the counter exceeds 1096 iterations (i.e. days):

```
j = 0
do while (RemainingGDDays > 0.1_dp)
    j = j + 1
```

```
    if (i > 365) then
        i = 1
    end if

    TDayMin_loc = GetTminCropReferenceRun_i(i)
    TDayMax_loc = GetTmaxCropReferenceRun_i(i)

    ! if there is a problem with the temperature data,
    ! this will prevent a permanent loop that will hang the program
    if (j > 1096) then
        write (*,*) 'ERROR: cannot determine crop maturity date'
        RemainingGDDays = 0.0_dp
    end if
end do
```

In South Africa, sugarcane can take up to 730 days (2 years) to reach physiological maturity due to the high number of required GDDs to reach physiological maturity. The counter limit of 1096 days represents a maximum season length of 3 years, which is more than sufficient for any crop.

The above code change represents a temporary fix to ensure the modified program produces identical results to v7.2. In a future release, it is recommended that the size of the *Tm**CropReferenceRun* array in *global.f90* is increased from 365 to 1096 days.

Step b: A division-by-zero error occurred in subroutine *DetermineBiomassAndYield*() in *simul.f90* when *SumKci* is divided by *SumKcTopStress*. This was fixed by adding the following code:

```
if (SumKcTopStress > 0._dp) then
    if (SumKci/SumKcTopStress < 1._dp) then
     .
    else
     .
    end if
end if
```

Again, this temporary fix needs further investigation to determine why *SumKcTopStress* is zero, so that the identified problem (i.e. bug) can be addressed in a future release.

Step c: Another division-by-zero error occurred on line 3817 in subroutine *DetermineCCiGDD*() in *simul.f90* during the calculation of *Wrelative* because *GetRootZoneWC_FC* - *GetRootZoneWC_WP* is zero. For example, AquaCrop crashed on AZ 1432 in the 49th season on day 35775 (1998/12/12) when both *GetRootZoneWC_FC*() and *GetRootZoneWC_WP*() are zero, thus causing a division-by-zero error. This was fixed by adding the following code:

```
if (GetRootZoneWC_FC() /= GetRootZoneWC_WP()) then
    Wrelative = (GetRootZoneWC_FC() - GetRootZoneWC_Actual()) &
                /(GetRootZoneWC_FC() - GetRootZoneWC_WP())
else
    Wrelative = (GetRootZoneWC_FC() - GetRootZoneWC_Actual())
end if
```

This temporary fix also requires further investigation to determine why the root zone water content (*RootZoneWC*) at field capacity (FC) and wilting point (WP) are both zero, thus resulting in the division-by-zero error. In subroutine *DetermineRootZoneWC*() in *global.f90*, *GetRootZoneWC_FC*() and *GetRootZoneWC_WP*() are both zero from day 35734 to 35775 because:
- *Factor* = 0, and
- *GetSoilLayer_GravelVol* = 0
for *GetCompartment_Layer*(compi) = 1 (where compi = 1)

Step d: Temperature files are no longer created in the **SIMUL** folder, since values are stored in global arrays for 365 days. During testing, the model attempted to access values beyond 365 days, thus creating an array-out-of-bound error. For example, in function *SeasonalSumOfKcPot*() in *global.f90*, if the day counter *j* exceeds 365 days, it is reset to 1 to re-use temperature values from the beginning of the array. Similarly, if the number of days of temperature data stored in the *TCropDotSIM* array is insufficient, the day counter is also reset to the beginning of the array.

```
j = j + 1
if (ReferenceClimate .eqv. .true.) then
    if (j > 365) then
        j = 1
        write(*,*)'ERROR: CropReferenceRun array too small'
        ! stop 1
    Endif

    Tndayi = GetTminCropReferenceRun_i(j)
    Txdayi = GetTmaxCropReferenceRun_i(j)
else
    if (j > TCropdotSIM(0,1)) j = 1

    ! use temperature values from TCrop.SIM
    ! this makes no sense since ReferenceClimate is always set to .true.
    Tndayi = TCropdotSIM(j,1)
    Txdayi = TCropdotSIM(j,2)
end if
```

As noted in Section 3.2.3 (cf. Step h), *ReferenceClimate* is always set to .true., and thus the above code can be simplified in a future release to not use data stored in the *TCropdotSIM array (which will cause v7.2 to crash as it did for v7.1)*.

Step e: In function *Bnormalized*() in *tempprocessing.f90*, similar code was added as mentioned above.

Step f: A similar change was made to function *SumCalendarDaysReferenceTnx*() in *preparefertilitysalinity.f90*, where the day counter *i* is reset to 1 if it exceeds 365 to prevent the following code generating an array-out-of-bounds error:

```
TDayMin_loc = GetTminCropReferenceRun_i(i)
TDayMax_loc = GetTmaxCropReferenceRun_i(i)
```

Step g: When testing the modified AquaCrop SA program, the model crashed in season 44 in AZ 1219 and in the 1st season in AZs 1432, 1463, 1464, 1471 and 1513. All of these crashes resulted from the same

error when accessing data for day 366 from the new *TCropDotSIM* array, which only has data for 365 days. To temporarily fix this issue, the following code was added to subroutine *TemperatureFileCoveringCropPeriod*() in tempprocessing.f90:

```
j = j + 1
if (j > 365) then
    write(*,*)'ERROR: Array TCropdotSIM is too small'
!   stop 1
else
    TCropdotSIM(j,1) = Tlow
    TCropdotSIM(j,2) = Thigh
end if
```

Regarding steps d to g above (which are related to step a), it is recommended that in a future release, the size of the *TCropdotSIM* array in *global.f90* is increased from 365 to 366 days, but preferably 1096 days (to accommodate crops like sugarcane, where season lengths can be up to two years in South Africa).

Summary: The above code changes were necessary to prevent AquaCrop hanging or crashing. This can result in some or all seasons not being simulated. For example, if the model is running a multiple project (**.PRM**) file with 49 consecutive seasons (for example) and crashes in the first season, no output will be created. This issue needs to be addressed by FAO in a future release. For AquaCrop SA v6.0, a temporary workaround involved running the model per season by creating individual single project (**.PRO**) files, instead of one larger **.PRM** file. The disadvantage of this workaround is it increases the overall model run time.

3.4 Other minor code changes

Other minor code changes were implemented as described next, which can be found by searching for "! RPK fix 25".

25) **Step a**: In subroutine *CreateTnXReferenceFile*() in *tempprocessing.f90*, the following code **cannot** be commented out, especially the first three lines:

```
i = len(trim(TemperatureFile))
TempString = trim(TemperatureFile)
call SetTnxReferenceFile(TempString(:i-4) // 'Reference.Tnx')
FullName = trim(GetPathNameSimul()) // GetTnxReferenceFile()
call SetTnxReferenceFileFull(FullName)
```

Variable *TnxReferenceFileFull*() represents the ***Reference.Tnx** file no longer created in the **SIMUL** folder. However, this variable is used elsewhere in the code, which needs to be removed before the above-mentioned code can be finally commented out. This should be addressed in a future release.

Step b: *In tempprocessing.f90*, the following subroutines are no longer needed, and thus were commented out (i.e. removed):

```
! *Reference.Tnx file no longer created
- subroutine fTnxReference_open()
- subroutine fTnxReference_write()
- subroutine fTnxReference_close()
- subroutine fTnxReference_erase() ! never called
```

```
! TnxReference365Days.SIM no longer created
- subroutine fTnxReference365Days_open()
- subroutine fTnxReference365Days_write()
- subroutine fTnxReference365Days_close()
- subroutine fTnxReference365Days_erase() ! never called
```

Step c: In subroutine *CreateTnxReferenceFile*() in *tempprocessing.f90*, there is no need to unlink the following files, as they are no longer created:

- ***Reference.Tnx** file
- **TnxReference365Days.SIM**
- **TCropReference.SIM**

No other calls to unlink files were found that needed to be commented out.

Step d: In subroutine *FinalizeTheProgram*() in *startunit.f90*, the file **AllDone.OUT** is no longer created in the **OUTP** folder. Instead, "All done" is written to the **ListProjectsLoaded.OUT** file, which helps reduce disk I/O.

Step e: In subroutine *WriteProjectsInfo*(line) in *startunit.f90*, the call to *fProjects_write*('') is incorrect, and should rather be *fProjects_write*(trim(line)). Instead, this subroutine was replaced with *fProjects_write*(), and thus is no longer needed (i.e. was commented out). The heading “Projects handled:” is now correctly written to the **ListProjectsLoaded.OUT** file.

Step f: To prevent possible memory leaks, allocated global variables were deallocated at the end of subroutine *RunSimulation*() in *run.f90*.

Step g: Finally, the version number of the modified program was changed from 7.2 to 7.3 in function *GetVersionString*() in *utils.f90*. Similarly, in function *GetReleaseDate*() in *utils.90*, the release date was changed from August 2024 to April 2025.

4 TESTING AND PERFORMANCE OF AQUACROP v7.3

4.1 Testing of v7.3

4.1.1 Test case

The modified version of the AquaCrop SA program (now called v7.3) produced identical results to the test case provided by FAO (cf. Section 2.3.1). This indicates that all of the above-mentioned bug fixes and performance improvements did not alter the program's ability to correctly simulate multiple cuttings of alfalfa grown in Ottawa (Canada) across three years.

4.1.2 Altitude Zones

For each AZ, a multiple project (**.PRM**) file was generated with the AquaCrop v7.2 GUI program for soybean planted on the 1st of November, with climate data spanning 50 years, thus producing up to 49 consecutive seasons. The **.PRM** file for each AZ was then compared to that generated with the GenPrm program and slight differences (of only 1 day) were noted for two AZs (0676 and 4490), as highlighted in Table 4-1. This resulted is minor differences in output, in particular for thermal time (6 GDDs for AZ 0676), which caused very small differences in soil water evaporation (E) of 1-2 mm. In addition, slight differences in output were also noticed for two other AZs (1219 and 1471). Overall, the modified version of the SA program (v7.3) produced almost identical results to the GUI program, since biomass, yield and crop water productivity were identical, except for the negligible difference shown in the table below for AZ 1471. All of the differences highlighted in Table 4-1 are attributed to rounding errors between the Pascal (GUI) and Fortran (SA and GenPrm) programming languages.

Table 4-1 Differences in simulations produced by AquaCrop GUI v7.2 compared to those produced by AquaCrop SA v7.3, with **.PRM** files created with the GenPrm program

AZ no.	Season	Variable	AquaCrop GUI v7.2	AquaCrop SA v7.3 & GenPrm	Absolute difference
0676	1956/57	Last day of simulation	20602 (1957/05/28)	20601 (1957/05/27)	1
		GDDs (°C.day)	2031	2025	6
		E (mm)	150	148	2
	1957/58	E (mm)	175	176	1
1219	1989/90	StoStr (%)	4	5	1
1471	1984/85	Tr/Trx (%)	65	64	1
		Biomass (dry t ha^{-1})	5.222	5.221	0.001
4490	1982/83	Last day of simulation	30099 (1983/05/29)	30100 (1983/05/30)	1
		GDDs (°C.day)	2025	2027	2
		E (mm)	299	300	1

4.2 Reduction in model run time

The GUI (v7.2) program was run for each of the 28 test AZs. It is important to note that AZs 1432, 1463, 1464, 1471 and 1513 (cf. Figure 2-1 in Section 2.1) are particularly cold and are therefore not suited to soybean production. Hence, only 18-39 (of 49) consecutive seasons were simulated by the model due to very long (economically unviable) season lengths. For example, only 18 seasons were simulated by the GUI program for AZ 1432, where crop cycle lengths range from 675-773 (average of 731) days. In the first season, the simulation starts on the 1st of November 1950 and ends on the 9th of January 1953 when the crop reaches physiological maturity after the accumulation of 2025 GDDs. Hence, the next season starts and ends on 1953/11/01 to 1956/01/10 respectively. However, the GenPrm program has the ability to generate **.PRM** files with seasons that start in each year, regardless of when the crop reaches physiological maturity. The number of simulations increases to 48, since the final season is from 1997/11/01 to 1999/11/26, and there is no climate data beyond 1999/12/31 to simulate the 49th season starting on 1999/11/01 and finishing in late 2001 or early 2002. Since the number of simulated seasons increases from 18 to 48, the average yield changes from 1.883 to 2.048 dry t ha^{-1}, with the latter value deemed a better reflection of the zone's suitability for soybean cultivation (since it is calculated from 30 additional seasons of data). The maximum yield increases from 2.906 dry t ha^{-1} (11th season; 1978/11/01-1980/11/03) to 4.242 dry t ha^{-1} (46th season; 1995/11/01-1998/02/04). FAO are therefore encouraged to modify the GUI program to allow users to simulate seasons that start in consecutive years, thus mimicking GenPrm's ability. Using the **.PRM** files created by GenPrm for 49 seasons (except for zone 1432), the GUI program was run again and timed using a hand-held stopwatch. The model run time was converted into seconds and rounded down, as shown in Table 4-2. Model run times ranged from 58-301 (average of 99) s. The performance tests were conducted on a Windows 10 PC (Intel core i9 7900X CPU; 64 GB of RAM; Seagate 8 TB 7 200 rpm hard drive), running Windows Subsystem for Linux (WSL v2).

Versions 6.0, 7.1 and 7.2 of the SA program were also run with the same **.PRM** files and timed using the Unix time command in WSL. Similarly, the modified program (v7.3) was also timed, but compiled in both debugging (v7.3d) and final release modes. From Table 4-2, it is clear that v6.0 of the SA program runs on average ~1.7 times faster than the GUI program (both written in Pascal). The decision by FAO to port the SA program from Pascal (v6) to Fortran (v7) resulted in a substantial increase in model performance, considering SA v7.1 runs 7.9 times faster than v6.0. The largest reduction in model run time occurs in AZ 1432, which decreased from 200 to 9 s (95.4% reduction). Version 7.2 of the SA program runs 27.9% slower than the previous version due to the changes briefly described in Section 2.3, since three extra files are created in the **SIMUL** folder (***Reference.Tnx**, **TCropReference.SIM** and **TnxReference365Days.SIM**). This explains why it was so important to reduce disk I/O when creating v7.3, as explained in Section 3.2. From Table 4-2, SA v7.3 now runs ~27 times faster than v7.2, since the model run time was reduced by 96% on average. The comparison excludes two AZs (0676 and 2680), where SA v7.1 and 7.2 crashes and produces no output due to the bug described in Section 3.1 (cf. point 9) due to single-word soil descriptions such as sand, silt and clay. It is worth noting that compiling SA v7.3 for debugging and development (v7.3d in Table 4-2) results in the program running 5.6 times slower, as expected.

Using the averaged model run time across 28 AZs, Table 4-3 shows that v7.3 of the SA program runs 259 times faster than the GUI (v7.2) program and ~160x faster than v6.0 (SA), both of which are written in Pascal. Similarly, v7.3 is ~20x and 25x faster than v7.1 and v7.2 respectively, which are coded in Fortran.

Table 4-2 Model run times when simulating 49 consecutive seasons from 1950/51 to 1998/99 using different versions of the AquaCrop program for each Altitude Zone (AZ)

AZ	Model run time (s)					
	GUI (v7.2)	SA (v6.0)	SA (v7.1)	SA (v7.2)	SA (v7.3)	SA (v7.3d)
0419	71	40.74	7.35	9.38	0.28	1.49
0423	67	39.98	7.25	9.29	0.28	1.37
0469	66	39.51	7.18	9.27	0.28	1.49
0524	67	38.11	7.16	9.22	0.27	1.29
0556	81	49.98	7.27	9.35	0.30	1.79
0563	70	41.54	7.17	9.30	0.29	1.59
0571	97	58.03	7.44	9.44	0.36	2.03
0643	66	38.81	7.15	9.34	0.31	1.31
0652	81	47.11	7.30	9.33	0.34	1.67
0676	90	54.45	crash	crash	0.33	1.80
0682	68	40.26	7.11	9.28	0.29	1.36
0733	93	56.14	7.34	10.39	0.38	1.99
0855	70	41.25	7.12	9.40	0.31	1.43
0899	77	46.76	7.18	9.28	0.33	1.59
0901	74	45.14	7.16	9.28	0.33	1.56
0952	68	40.07	8.34	9.39	0.31	1.39
1081	77	46.35	8.01	10.03	0.35	1.77
1207	92	56.35	7.92	9.39	0.37	1.96
1219	137	85.66	7.73	9.85	0.50	3.02
1265	73	42.12	7.09	9.25	0.28	1.52
1432*	301	200.15	9.11	11.33	1.05	8.72
1463	157	99.29	7.94	10.14	0.53	3.96
1464	145	92.02	7.82	10.03	0.51	3.69
1471	165	106.84	8.04	10.22	0.56	4.12
1513	159	102.57	8.97	10.36	0.55	4.02
2680	74	43.78	crash	crash	0.30	1.51
4490	117	71.86	7.54	9.74	0.39	2.52
4697	58	34.48	7.08	9.79	0.27	1.35
ave	**99**	**60.69**	**7.57**	**9.66**	**0.38**	**2.26**
min	**58**	**34.48**	**7.08**	**9.22**	**0.27**	**1.29**
max	**301**	**200.15**	**9.11**	**11.33**	**1.05**	**8.72**

*48 not 49 seasons

Table 4-3 Performance improvement of AquaCrop SA v7.3 relative to other versions of the model

	GUI v7.2	SA v6.0	SA v7.1	SA v7.2	SA v7.3
GUI v7.2	**1.0**				
SA v6.0	1.6	**1.0**			
SA v7.1	13.0	8.0	**1.0**		
SA v7.2	10.2	6.3	0.8	**1.0**	
SA v7.3	259.2	159.6	19.9	25.4	**1.0**

REFERENCES

FAO (FOOD AND AGRICULTURAL ORGANISATION) (2024) *AquaCrop Version 7.2 Release Note (August 2024)*. Food And Agricultural Organisation, Rome, Italy. https://github.com/KUL-RSDA/AquaCrop/releases

GEERTS D, RAES D, GARCIA M, MIRANDA R, CUSICANQUI JA, TABOADA C, MENDOZA J, HUANCA R, MAMANI A, CONDORI O, MAMANI J, MORALES B, OSCO V, STEDUTO P (2009) Simulating yield response of quinoa to water availability with AquaCrop. Agronomy Journal **101 (3)** 499-508.

HENG LK, HSIAO TC, EVETT SR, HOWELL TA, STEDUTO P (2009) Testing of FAO AquaCrop model for rainfed and irrigated maize. Agronomy Journal **101 (3)** 488-498.

HSIAO TC, LEE H, STEDUTO P, BASILIO RL, RAES D, FERERES E (2009) AquaCrop – the FAO crop model to simulate yield response to water: III. Parameterization and testing for maize. Agronomy Journal **101 (3)** 448-459.

KUNZ RP, DAVIS NS, THORNTON-DIBB SLC, STEYN JM, DU TOIT ES, JEWITT GPW (2015) *Assessment of biofuel feedstock production in South Africa: Atlas of water use and yield of biofuel crops in suitable growing areas (Volume 3*). WRC Report No. TT 652/15, Water Research Commission (WRC), Pretoria, South Africa.

KUNZ R, MASANGANISE J, REDDY K, MABHAUDHI T, LEMBEDE L, NAIKEN V, FERRER S (2020) *Water use and yield of soybean and grain sorghum for biofuel production*. WRC Report No. 2491/1/20, Water Research Commission (WRC), Pretoria, South Africa. 361 pp.

KUNZ R, MABHAUDHI T (2023) *Volume 2: Climate change atlas for rainfed production of selected underutilised crops*. WRC Report No. 2717/2/23, Water Research Commission (WRC), Pretoria, South Africa. 145 pp.

KUNZ R, REDDY K, MTHEMBU T, LAKE S, CHIMONYO V, MABHAUDHI T (2024) *Crop and nutritional water productivity of sweet potato and taro*. WRC Report No. 3124/1/24, Water Research Commission (WRC), Pretoria, South Africa.

LAKE S (2024) *A novel approach to mapping areas suitable for rainfed production of bambara nut and cowpea*. Unpublished MSc dissertation, Centre for Water Resources Research, School of Agricultural, Earth and Environmental Sciences, University of KwaZulu-Natal, Pietermaritzburg, South Africa.

KUNZ RP, SCHULZE RE (2017) Grain Sorghum Production in South Africa and Climate Change, In: Schulze RE (Ed.), *Handbook for Farmers, Officials and Other Stakeholders on Adaptation to Climate Change in the Agriculture Sector within South Africa*, Chapter C5, 282-294. University of KwaZulu-Natal, Pietermaritzburg, South Africa.

MABHAUDHI T, KUNZ RP, SCHULZE RE (2017a) Taro (Amadumbe) in South Africa and Climate Change, In: Schulze RE (Ed.), *Handbook for Farmers, Officials and Other Stakeholders on Adaptation to Climate Change in the Agriculture Sector within South Africa*, Chapter C6, 295-300. University of KwaZulu-Natal, Pietermaritzburg, South Africa.

MABHAUDHI T, KUNZ RP, SCHULZE RE (2017b) Bambara Groundnut in South Africa and Climate Change, In: Schulze RE (Ed.), *Handbook for Farmers, Officials and Other Stakeholders on Adaptation to Climate Change in the Agriculture Sector within South Africa*, Chapter C7, 301-306. University of KwaZulu-Natal, Pietermaritzburg, South Africa.

RAES D, STEDUTO P, HSIAO TC, FERERES E (2009) AquaCrop – the FAO crop model to simulate yield response to water: II. Main algorithms and software description. Agronomy Journal **101 (3)** 438-447.

SCHULZE RE (2017) *Handbook for Farmers, Officials and Others on Adaptation to Climate Change in the Agricultural Sector of South Africa*. Department of Agriculture, Forestry and Fisheries, Pretoria, South Africa.

STEDUTO P, HSIAO TC, RAES D, FERERES E (2009) AquaCrop – the FAO crop model to simulate yield response to water: I. Concepts and underlying principles. Agronomy Journal **101 (3)** 426-437.