Software Requirements Specification for Projectile

Samuel J. Crawford, Brooks MacLachlan, and W. Spencer Smith

July 29, 2021

Contents

1 Reference Material

This section records information for easy reference.

1.1 Table of Units

The unit system used throughout is SI (Système International d'Unités). In addition to the basic units, several derived units are also used. For each unit, the Table of Units lists the symbol, a description and the SI name.

Symbol	Description	SI Name
m	length	metre
rad	angle	radian
s	time	second

Table 1: Table of Units

1.2 Table of Symbols

The symbols used in this document are summarized in the Table of Symbols along with their units. Throughout the document, symbols in bold will represent vectors, and scalars otherwise. The symbols are listed in alphabetical order. For vector quantities, the units shown are for each component of the vector.

Symbol	Description	Units
a	Scalar acceleration	$\frac{\mathrm{m}}{\mathrm{s}^2}$
a^c	Constant acceleration	$\frac{\mathrm{m}}{\mathrm{s}^2}$
a_{x}	x-component of acceleration	$\frac{\mathrm{m}}{\mathrm{s}^2}$
${a_{\mathrm{x}}}^{\mathrm{c}}$	x-component of constant acceleration	$\frac{\mathrm{m}}{\mathrm{s}^2}$
a_{y}	y-component of acceleration	$\frac{\mathrm{m}}{\mathrm{s}^2}$
${a_{\mathrm{y}}}^{\mathrm{c}}$	y-component of constant acceleration	$\frac{\mathrm{m}}{\mathrm{s}^2}$
$\mathbf{a}$	Acceleration	$\frac{\mathrm{m}}{\mathrm{s}^2}$
$\mathbf{a}^{\mathrm{c}}$	Constant acceleration vector	$\frac{\mathrm{m}}{\mathrm{s}^2}$
d_{offset}	Distance between the target position and the landing position	m
$\mathbf{g}$	Gravitational acceleration	$\frac{\mathrm{m}}{\mathrm{s}^2}$
p	Scalar position	m
p^{i}	Initial position	m
p_{land}	Landing position	m
p_{target}	Target position	m

Symbol	Description	Units
p_{x}	x-component of position	m
$p_{\text{x}}{}^{\text{i}}$	x-component of initial position	m
p_{y}	y-component of position	m
$p_{\text{y}}{}^{\text{i}}$	y-component of initial position	m
$\mathbf{p}$	Position	m
s	Output message as a string	–
t	Time	s
t_{flight}	Flight duration	s
v	Speed	$\frac{\text{m}}{\text{s}}$
v^{i}	Initial speed	$\frac{\text{m}}{\text{s}}$
v_{launch}	Launch speed	$\frac{\text{m}}{\text{s}}$
v_{x}	x-component of velocity	$\frac{\text{m}}{\text{s}}$
$v_{\text{x}}{}^{\text{i}}$	x-component of initial velocity	$\frac{\text{m}}{\text{s}}$
v_{y}	y-component of velocity	$\frac{\text{m}}{\text{s}}$
$v_{\text{y}}{}^{\text{i}}$	y-component of initial velocity	$\frac{\text{m}}{\text{s}}$
$\mathbf{v}$	Velocity	$\frac{\text{m}}{\text{s}}$
$\mathbf{v}(t)$	1D speed	$\frac{\text{m}}{\text{s}}$
$\mathbf{v}^{\text{i}}$	Initial velocity	$\frac{\text{m}}{\text{s}}$
ε	Hit tolerance	–
θ	Launch angle	rad
π	Ratio of circumference to diameter for any circle	–

Table 2: Table of Symbols

1.3 Abbreviations and Acronyms

Abbreviation	Full Form
1D	One-Dimensional
2D	Two-Dimensional
A	Assumption
DD	Data Definition
GD	General Definition
GS	Goal Statement
IM	Instance Model

Abbreviation	Full Form
PS	Physical System Description
R	Requirement
SRS	Software Requirements Specification
TM	Theoretical Model
Uncert.	Typical Uncertainty

Table 3: Abbreviations and Acronyms

2 Introduction

Projectile motion is a common problem in physics. Therefore, it is useful to have a program to solve and model these types of problems. The program documented here is called Projectile.

The following section provides an overview of the Software Requirements Specification (SRS) for Projectile. This section explains the purpose of this document, the scope of the requirements, the characteristics of the intended reader, and the organization of the document.

2.1 Scope of Requirements

The scope of the requirements includes the analysis of a two-dimensional (2D) projectile motion problem with constant acceleration.

3 Specific System Description

This section first presents the problem description, which gives a high-level view of the problem to be solved. This is followed by the solution characteristics specification, which presents the assumptions, theories, and definitions that are used.

3.1 Problem Description

A system is needed to efficiently and correctly predict the landing position of a projectile.

3.1.1 Terminology and Definitions

This subsection provides a list of terms that are used in the subsequent sections and their meaning, with the purpose of reducing ambiguity and making it easier to correctly understand the requirements.

- Launcher: Where the projectile is launched from and the device that does the launching.

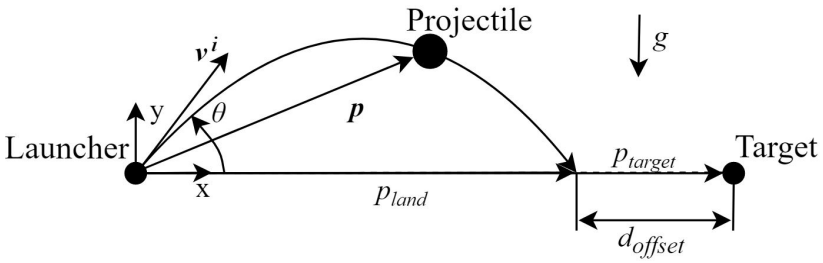

Figure 1: The physical system

- Projectile: The object to be launched at the target.
- Target: Where the projectile should be launched to.
- Gravity: The force that attracts one physical body with mass to another.
- Cartesian coordinate system: A coordinate system that specifies each point uniquely in a plane by a set of numerical coordinates, which are the signed distances to the point from two fixed perpendicular oriented lines, measured in the same unit of length (from [2]).
- Rectilinear: Occuring in one dimension.

3.1.2 Physical System Description

The physical system of Projectile, as shown in Fig:Launch, includes the following elements:

PS1: The launcher.

PS2: The projectile (with initial velocity $\mathbf{v}^{\mathrm{i}}$ and launch angle θ).

PS3: The target.

3.1.3 Goal Statements

Given the initial velocity vector of the projectile, the goal statements are:

targetHit: Determine if the projectile hits the target.

3.2 Solution Characteristics Specification

The instance models that govern Projectile are presented in the Instance Model Section. The information to understand the meaning of the instance models and their derivation is also presented, so that the instance models can be verified.

3.2.1 Assumptions

This section simplifies the original problem and helps in developing the theoretical models by filling in the missing information for the physical system. The assumptions refine the scope by providing more detail.

twoDMotion: The projectile motion is two-dimensional (2D). (RefBy: GD:velVec and GD:posVec.)

cartSyst: A Cartesian coordinate system is used (from A:neglectCurv). (RefBy: GD:velVec and GD:posVec.)

yAxisGravity: The direction of the y-axis is directed opposite to gravity. (RefBy: IM:calOfLandingDist, IM:calOfLandingTime, and A:accelYGravity.)

launchOrigin: The launcher is coincident with the origin. (RefBy: IM:calOfLandingDist and IM:calOfLandingTime.)

targetXAxis: The target lies on the x-axis (from A:neglectCurv). (RefBy: IM:calOfLandingTime.)

posXDirection: The positive x-direction is from the launcher to the target. (RefBy: IM:offsetIM, IM:messageIM, IM:calOfLandingDist, and IM:calOfLandingTime.)

constAccel: The acceleration is constant (from A:accelXZero, A:accelYGravity, A:neglectDrag, and A:freeFlight). (RefBy: GD:velVec and GD:posVec.)

accelXZero: The acceleration in the x-direction is zero. (RefBy: IM:calOfLandingDist and A:constAccel.)

accelYGravity: The acceleration in the y-direction is the acceleration due to gravity (from A:yAxisGravity). (RefBy: IM:calOfLandingTime and A:constAccel.)

neglectDrag: Air drag is neglected. (RefBy: A:constAccel.)

pointMass: The size and shape of the projectile are negligible, so that it can be modelled as a point mass. (RefBy: GD:rectPos and GD:rectVel.)

freeFlight: The flight is free; there are no collisions during the trajectory of the projectile. (RefBy: A:constAccel.)

neglectCurv: The distance is small enough that the curvature of the Earth can be neglected. (RefBy: A:targetXAxis and A:cartSyst.)

timeStartZero: Time starts at zero. (RefBy: GD:velVec, GD:rectPos, GD:rectVel, GD:posVec, and IM:calOfLandingTime.)

gravAccelValue: The acceleration due to gravity is assumed to have the value provided in the section for Values of Auxiliary Constants. (RefBy: IM:calOfLandingDist and IM:calOfLandingTime.)

3.2.2 Theoretical Models

This section focuses on the general equations and laws that Projectile is based on.

Refname	**TM:acceleration**
Label	Acceleration
Equation	$$\mathbf{a} = \frac{d\mathbf{v}}{dt}$$
Description	$\mathbf{a}$ is the acceleration ($\frac{\text{m}}{\text{s}^2}$) t is the time (s) $\mathbf{v}$ is the velocity ($\frac{\text{m}}{\text{s}}$)
Source	[1]
RefBy	GD:rectVel

Refname	**TM:velocity**
Label	Velocity
Equation	$$\mathbf{v} = \frac{d\mathbf{p}}{dt}$$
Description	$\mathbf{v}$ is the velocity ($\frac{\text{m}}{\text{s}}$) t is the time (s) $\mathbf{p}$ is the position (m)
Source	[3]
RefBy	GD:rectPos

3.2.3 General Definitions

This section collects the laws and equations that will be used to build the instance models.

Refname	**GD:rectVel**
Label	Rectilinear (1D) velocity as a function of time for constant acceleration
Units	$\frac{\text{m}}{\text{s}}$
Equation	$$\mathbf{v}(t) = v^{\text{i}} + a^c t$$
Description	$\mathbf{v}(t)$ is the 1D speed ($\frac{\text{m}}{\text{s}}$) v^{i} is the initial speed ($\frac{\text{m}}{\text{s}}$) a^c is the constant acceleration ($\frac{\text{m}}{\text{s}^2}$) t is the time (s)
Source	[4, (pg. 8)]
RefBy	GD:velVec and GD:rectPos

Detailed derivation of rectilinear velocity: Assume we have rectilinear motion of a particle (of negligible size and shape, from A:pointMass); that is, motion in a straight line. The velocity is v and the acceleration is a. The motion in TM:acceleration is now one-dimensional with a constant acceleration, represented by a^c. The initial velocity (at $t = 0$, from A:timeStartZero) is represented by v^{i}. From TM:acceleration, using the above symbols we have:

$$a^c = \frac{dv}{dt}$$

Rearranging and integrating, we have:

$$\int_{v^{\text{i}}}^{v} 1 \, dv = \int_0^t a^c \, dt$$

Performing the integration, we have the required equation:

$$\mathbf{v}(t) = v^{\mathrm{i}} + a^c t$$

Refname	**GD:rectPos**
Label	Rectilinear (1D) position as a function of time for constant acceleration
Units	m
Equation	$$p = p^{\mathrm{i}} + v^{\mathrm{i}} t + \frac{a^c t^2}{2}$$
Description	p is the scalar position (m) p^{i} is the initial position (m) v^{i} is the initial speed ($\frac{\mathrm{m}}{\mathrm{s}}$) t is the time (s) a^c is the constant acceleration ($\frac{\mathrm{m}}{\mathrm{s}^2}$)
Source	[4, (pg. 8)]
RefBy	GD:posVec

Detailed derivation of rectilinear position: Assume we have rectilinear motion of a particle (of negligible size and shape, from A:pointMass); that is, motion in a straight line. The position is p and the velocity is v. The motion in TM:velocity is now one-dimensional. The initial position (at $t = 0$, from A:timeStartZero) is represented by p^{i}. From TM:velocity, using the above symbols we have:

$$v = \frac{dp}{dt}$$

Rearranging and integrating, we have:

$$\int_{p^{\mathrm{i}}}^{p} 1\,dp = \int_{0}^{t} v\,dt$$

From GD:rectVel, we can replace v:

$$\int_{p^{\mathrm{i}}}^{p} 1\,dp = \int_{0}^{t} v^{\mathrm{i}} + a^{c}t\,dt$$

Performing the integration, we have the required equation:

$$p = p^{\mathrm{i}} + v^{\mathrm{i}}t + \frac{a^{c}t^{2}}{2}$$

Refname	**GD:velVec**
Label	Velocity vector as a function of time for 2D motion under constant acceleration
Units	$\frac{\mathrm{m}}{\mathrm{s}}$
Equation	$\mathbf{v} = \begin{bmatrix} {v_{\mathrm{x}}}^{\mathrm{i}} + {a_{\mathrm{x}}}^{\mathrm{c}}t \\ {v_{\mathrm{y}}}^{\mathrm{i}} + {a_{\mathrm{y}}}^{\mathrm{c}}t \end{bmatrix}$
Description	$\mathbf{v}$ is the velocity ($\frac{\mathrm{m}}{\mathrm{s}}$) ${v_{\mathrm{x}}}^{\mathrm{i}}$ is the x-component of initial velocity ($\frac{\mathrm{m}}{\mathrm{s}}$) ${a_{\mathrm{x}}}^{\mathrm{c}}$ is the x-component of constant acceleration ($\frac{\mathrm{m}}{\mathrm{s}^2}$) t is the time (s) ${v_{\mathrm{y}}}^{\mathrm{i}}$ is the y-component of initial velocity ($\frac{\mathrm{m}}{\mathrm{s}}$) ${a_{\mathrm{y}}}^{\mathrm{c}}$ is the y-component of constant acceleration ($\frac{\mathrm{m}}{\mathrm{s}^2}$)
Source	–
RefBy	

Detailed derivation of velocity vector: For a two-dimensional Cartesian coordinate system (A:twoDMotion and A:cartSyst), we can represent the velocity vector as $\mathbf{v} = \begin{bmatrix} v_\mathrm{x} \\ v_\mathrm{y} \end{bmatrix}$ and the acceleration vector as $\mathbf{a} = \begin{bmatrix} a_\mathrm{x} \\ a_\mathrm{y} \end{bmatrix}$. The acceleration is assumed to be constant (A:constAccel) and the constant acceleration vector is represented as $\mathbf{a}^\mathrm{c} = \begin{bmatrix} {a_\mathrm{x}}^\mathrm{c} \\ {a_\mathrm{y}}^\mathrm{c} \end{bmatrix}$. The initial velocity (at $t = 0$, from A:timeStartZero) is represented by $\mathbf{v}^\mathrm{i} = \begin{bmatrix} {v_\mathrm{x}}^\mathrm{i} \\ {v_\mathrm{y}}^\mathrm{i} \end{bmatrix}$. Since we have a Cartesian coordinate system, GD:rectVel can be applied to each coordinate of the velocity vector to yield the required equation:

$$\mathbf{v} = \begin{bmatrix} {v_\mathrm{x}}^\mathrm{i} + {a_\mathrm{x}}^\mathrm{c} t \\ {v_\mathrm{y}}^\mathrm{i} + {a_\mathrm{y}}^\mathrm{c} t \end{bmatrix}$$

Refname	**GD:posVec**
Label	Position vector as a function of time for 2D motion under constant acceleration
Units	m
Equation	$$\mathbf{p} = \begin{bmatrix} {p_\text{x}}^\text{i} + {v_\text{x}}^\text{i} t + \frac{{a_\text{x}}^\text{c} t^2}{2} \\ {p_\text{y}}^\text{i} + {v_\text{y}}^\text{i} t + \frac{{a_\text{y}}^\text{c} t^2}{2} \end{bmatrix}$$
Description	$\mathbf{p}$ is the position (m) ${p_\text{x}}^\text{i}$ is the x-component of initial position (m) ${v_\text{x}}^\text{i}$ is the x-component of initial velocity ($\frac{\text{m}}{\text{s}}$) t is the time (s) ${a_\text{x}}^\text{c}$ is the x-component of constant acceleration ($\frac{\text{m}}{\text{s}^2}$) ${p_\text{y}}^\text{i}$ is the y-component of initial position (m) ${v_\text{y}}^\text{i}$ is the y-component of initial velocity ($\frac{\text{m}}{\text{s}}$) ${a_\text{y}}^\text{c}$ is the y-component of constant acceleration ($\frac{\text{m}}{\text{s}^2}$)
Source	–
RefBy	IM:calOfLandingDist and IM:calOfLandingTime

Detailed derivation of position vector: For a two-dimensional Cartesian coordinate system (A:twoDMotion and A:cartSyst), we can represent the position vector as $\mathbf{p} = \begin{bmatrix} p_\text{x} \\ p_\text{y} \end{bmatrix}$, the velocity vector as $\mathbf{v} = \begin{bmatrix} v_\text{x} \\ v_\text{y} \end{bmatrix}$, and the acceleration vector as $\mathbf{a} = \begin{bmatrix} a_\text{x} \\ a_\text{y} \end{bmatrix}$. The acceleration is assumed to be constant (A:constAccel) and the constant acceleration vector is represented as $\mathbf{a}^\text{c} = \begin{bmatrix} {a_\text{x}}^\text{c} \\ {a_\text{y}}^\text{c} \end{bmatrix}$. The initial velocity (at $t = 0$, from A:timeStartZero) is represented by

$\mathbf{v}^{\mathrm{i}} = \begin{bmatrix} {v_{\mathrm{x}}}^{\mathrm{i}} \\ {v_{\mathrm{y}}}^{\mathrm{i}} \end{bmatrix}$. Since we have a Cartesian coordinate system, GD:rectPos can be applied to each coordinate of the position vector to yield the required equation:

$$\mathbf{p} = \begin{bmatrix} {p_{\mathrm{x}}}^{\mathrm{i}} + {v_{\mathrm{x}}}^{\mathrm{i}} t + \frac{{a_{\mathrm{x}}}^{\mathrm{c}} t^2}{2} \\ {p_{\mathrm{y}}}^{\mathrm{i}} + {v_{\mathrm{y}}}^{\mathrm{i}} t + \frac{{a_{\mathrm{y}}}^{\mathrm{c}} t^2}{2} \end{bmatrix}$$

3.2.4 Data Definitions

This section collects and defines all the data needed to build the instance models.

Refname	**DD:vecMag**
Label	Speed
Symbol	v
Units	$\frac{\text{m}}{\text{s}}$
Equation	$$v = \|\mathbf{v}\|$$
Description	v is the speed ($\frac{\text{m}}{\text{s}}$) $\mathbf{v}$ is the velocity ($\frac{\text{m}}{\text{s}}$)
Notes	For a given velocity vector $\mathbf{v}$, the magnitude of the vector ($\|\mathbf{v}\|$) is the scalar called speed.
Source	–
RefBy	DD:speedIY and DD:speedIX

Refname	**DD:speedIX**
Label	x-component of initial velocity
Symbol	${v_{\text{x}}}^{\text{i}}$
Units	$\frac{\text{m}}{\text{s}}$
Equation	$${v_{\text{x}}}^{\text{i}} = v^{\text{i}} \cos(\theta)$$
Description	${v_{\text{x}}}^{\text{i}}$ is the x-component of initial velocity ($\frac{\text{m}}{\text{s}}$) v^{i} is the initial speed ($\frac{\text{m}}{\text{s}}$) θ is the launch angle (rad)
Notes	v^{i} is from DD:vecMag. θ is shown in Fig:Launch.
Source	–
RefBy	IM:calOfLandingDist

Refname	**DD:speedIY**
Label	y-component of initial velocity
Symbol	${v_y}^{\mathrm{i}}$
Units	$\frac{\mathrm{m}}{\mathrm{s}}$
Equation	$${v_y}^{\mathrm{i}} = v^{\mathrm{i}} \sin(\theta)$$
Description	${v_y}^{\mathrm{i}}$ is the y-component of initial velocity ($\frac{\mathrm{m}}{\mathrm{s}}$) v^{i} is the initial speed ($\frac{\mathrm{m}}{\mathrm{s}}$) θ is the launch angle (rad)
Notes	v^{i} is from DD:vecMag. θ is shown in Fig:Launch.
Source	–
RefBy	IM:calOfLandingTime

3.2.5 Instance Models

This section transforms the problem defined in the problem description into one which is expressed in mathematical terms. It uses concrete symbols defined in the data definitions to replace the abstract symbols in the models identified in theoretical models and general definitions.

Refname	**IM:calOfLandingTime**
Label	Calculation of landing time
Input	v_{launch}, θ
Output	t_{flight}
Input Constraints	$v_{\text{launch}} > 0$ $0 < \theta < \frac{\pi}{2}$
Output Constraints	$t_{\text{flight}} > 0$
Equation	$t_{\text{flight}} = \frac{2 v_{\text{launch}} \sin(\theta)}{\mathbf{g}}$
Description	t_{flight} is the flight duration (s) v_{launch} is the launch speed ($\frac{\text{m}}{\text{s}}$) θ is the launch angle (rad) $\mathbf{g}$ is the gravitational acceleration ($\frac{\text{m}}{\text{s}^2}$)
Notes	The constraint $0 < \theta < \frac{\pi}{2}$ is from A:posXDirection and A:yAxisGravity, and is shown in Fig:Launch. $\mathbf{g}$ is defined in A:gravAccelValue. The constraint $t_{\text{flight}} > 0$ is from A:timeStartZero.
Source	–
RefBy	IM:calOfLandingDist and FR:Calculate-Values

Detailed derivation of flight duration: We know that ${p_y}^{\text{i}} = 0$ (A:launchOrigin) and ${a_y}^{\text{c}} = -\mathbf{g}$ (A:accelYGravity). Substituting these values into the y-direction of GD:posVec gives us:

$$p_y = {v_y}^{\text{i}} t - \frac{\mathbf{g}t^2}{2}$$

To find the time that the projectile lands, we want to find the t value (t_{flight}) where $p_y = 0$ (since the target is on the x-axis from A:targetXAxis). From the equation above we get:

$${v_y}^{\text{i}} t_{\text{flight}} - \frac{\mathbf{g}{t_{\text{flight}}}^2}{2} = 0$$

Dividing by t_{flight} (with the constraint $t_{\text{flight}} > 0$) gives us:

$${v_y}^{\text{i}} - \frac{\mathbf{g}t_{\text{flight}}}{2} = 0$$

Solving for t_{flight} gives us:

$$t_{\text{flight}} = \frac{2{v_y}^{\text{i}}}{\mathbf{g}}$$

From DD:speedIY (with $v^{\text{i}} = v_{\text{launch}}$) we can replace ${v_y}^{\text{i}}$:

$$t_{\text{flight}} = \frac{2v_{\text{launch}} \sin(\theta)}{\mathbf{g}}$$

Refname	**IM:calOfLandingDist**
Label	Calculation of landing position
Input	v_{launch}, θ
Output	p_{land}
Input Constraints	$v_{\text{launch}} > 0$ $0 < \theta < \frac{\pi}{2}$
Output Constraints	$p_{\text{land}} > 0$
Equation	$p_{\text{land}} = \frac{2{v_{\text{launch}}}^2 \sin(\theta) \cos(\theta)}{\mathbf{g}}$
Description	p_{land} is the landing position (m) v_{launch} is the launch speed ($\frac{\text{m}}{\text{s}}$) θ is the launch angle (rad) $\mathbf{g}$ is the gravitational acceleration ($\frac{\text{m}}{\text{s}^2}$)
Notes	The constraint $0 < \theta < \frac{\pi}{2}$ is from A:posXDirection and A:yAxisGravity, and is shown in Fig:Launch. $\mathbf{g}$ is defined in A:gravAccelValue. The constraint $p_{\text{land}} > 0$ is from A:posXDirection.
Source	–
RefBy	IM:offsetIM and FR:Calculate-Values

Detailed derivation of landing position: We know that ${p_x}^{\text{i}} = 0$ (A:launchOrigin) and ${a_x}^{\text{c}} = 0$ (A:accelXZero). Substituting these values into the x-direction of GD:posVec gives us:

$$p_x = {v_x}^{\text{i}} t$$

To find the landing position, we want to find the p_x value (p_{land}) at flight duration (from IM:calOfLandingTime):

$$p_{\text{land}} = \frac{{v_x}^{\text{i}} \cdot 2v_{\text{launch}} \sin(\theta)}{\mathbf{g}}$$

From DD:speedIX (with $v^{\text{i}} = v_{\text{launch}}$) we can replace ${v_x}^{\text{i}}$:

$$p_{\text{land}} = \frac{v_{\text{launch}} \cos(\theta) \cdot 2v_{\text{launch}} \sin(\theta)}{\mathbf{g}}$$

Rearranging this gives us the required equation:

$$p_{\text{land}} = \frac{2{v_{\text{launch}}}^2 \sin(\theta) \cos(\theta)}{\mathbf{g}}$$

Refname	**IM:offsetIM**
Label	Offset
Input	p_{land}, p_{target}
Output	d_{offset}
Input Constraints	$p_{\text{land}} > 0$ $p_{\text{target}} > 0$
Output Constraints	
Equation	$d_{\text{offset}} = p_{\text{land}} - p_{\text{target}}$
Description	d_{offset} is the distance between the target position and the landing position (m) p_{land} is the landing position (m) p_{target} is the target position (m)
Notes	p_{land} is from IM:calOfLandingDist. The constraints $p_{\text{land}} > 0$ and $p_{\text{target}} > 0$ are from A:posXDirection.
Source	–
RefBy	FR:Output-Values, IM:messageIM, and FR:Calculate-Values

Refname	**IM:messageIM**
Label	Output message
Input	d_{offset}, p_{target}
Output	s
Input Constraints	$d_{\text{offset}} > -p_{\text{land}}$ $p_{\text{target}} > 0$
Output Constraints	
Equation	$s = \begin{cases} \text{“The target was hit.”}, & \lvert\frac{d_{\text{offset}}}{p_{\text{target}}}\rvert < \varepsilon \\ \text{“The projectile fell short.”}, & d_{\text{offset}} < 0 \\ \text{“The projectile went long.”}, & d_{\text{offset}} > 0 \end{cases}$
Description	s is the output message as a string (Unitless) d_{offset} is the distance between the target position and the landing position (m) p_{target} is the target position (m) ε is the hit tolerance (Unitless)
Notes	d_{offset} is from IM:offsetIM. The constraint $p_{\text{target}} > 0$ is from A:posXDirection. The constraint $d_{\text{offset}} > -p_{\text{land}}$ is from the fact that $p_{\text{land}} > 0$, from A:posXDirection. ε is defined in Sec:Values of Auxiliary Constants.
Source	–
RefBy	FR:Output-Values and FR:Calculate-Values

3.2.6 Data Constraints

The Data Constraints Table shows the data constraints on the input variables. The column for physical constraints gives the physical limitations on the range of values that can be taken by the variable. The uncertainty column provides an estimate of the confidence with which the physical quantities can be measured. This information would be part of the input if one were performing an uncertainty quantification exercise. The constraints are conservative, to give the user of the model the flexibility to experiment with unusual situations. The column of typical values is intended to provide a feel for a common scenario.

Var	Physical Constraints	Typical Value	Uncert.
p_{target}	$p_{\text{target}} > 0$	1000 m	10%
v_{launch}	$v_{\text{launch}} > 0$	100 $\frac{\text{m}}{\text{s}}$	10%
θ	$0 < \theta < \frac{\pi}{2}$	$\frac{\pi}{4}$ rad	10%

Table 4: Input Data Constraints

3.2.7 Properties of a Correct Solution

The Data Constraints Table shows the data constraints on the output variables. The column for physical constraints gives the physical limitations on the range of values that can be taken by the variable.

Var	Physical Constraints
p_{land}	$p_{\text{land}} > 0$
d_{offset}	$d_{\text{offset}} > -p_{\text{land}}$

Table 5: Output Data Constraints

4 Requirements

This section provides the functional requirements, the tasks and behaviours that the software is expected to complete, and the non-functional requirements, the qualities that the software is expected to exhibit.

4.1 Functional Requirements

This section provides the functional requirements, the tasks and behaviours that the software is expected to complete.

Input-Values: Input the values from Tab:ReqInputs.

erify-Input-Values: Check the entered input values to ensure that they do not exceed the data constraints. If any of the input values are out of bounds, an error message is displayed and the calculations stop.

Calculate-Values: Calculate the following values: t_{flight} (from IM:calOfLandingTime), p_{land} (from IM:calOfLandingDist), d_{offset} (from IM:offsetIM), and s (from IM:messageIM).

Output-Values: Output s (from IM:messageIM) and d_{offset} (from IM:offsetIM).

Symbol	Description	Units
p_{target}	Target position	m
v_{launch}	Launch speed	$\frac{\text{m}}{\text{s}}$
θ	Launch angle	rad

Table 6: Required Inputs following FR:Input-Values

4.2 Non-Functional Requirements

This section provides the non-functional requirements, the qualities that the software is expected to exhibit.

Correct: The outputs of the code have the properties described in Properties of a Correct Solution.

Verifiable: The code is tested with complete verification and validation plan.

Understandable: The code is modularized with complete module guide and module interface specification.

Reusable: The code is modularized.

Maintainable: The traceability between requirements, assumptions, theoretical models, general definitions, data definitions, instance models, likely changes, unlikely changes, and modules is completely recorded in traceability matrices in the SRS and module guide.

Portable: The code is able to be run in different environments.

5 Traceability Matrices and Graphs

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed. Every time a component is changed, the items in the column of that component that are marked with an "X" should be modified as well. Tab:TraceMatAvsA shows the dependencies of assumptions on the assumptions.

Tab:TraceMatAvsAll shows the dependencies of data definitions, theoretical models, general definitions, instance models, requirements, likely changes, and unlikely changes on the assumptions. Tab:TraceMatRefvsRef shows the dependencies of data definitions, theoretical models, general definitions, and instance models with each other. Tab:TraceMatAllvsR shows the dependencies of requirements, goal statements on the data definitions, theoretical models, general definitions, and instance models.

	A:twoDMotion	**A:cartSyst**	**A:yAxisGravity**	**A:launchOrigin**	**A:target**
A:twoDMotion					
A:cartSyst					
A:yAxisGravity					
A:launchOrigin					
A:targetXAxis					
A:posXDirection					
A:constAccel					
A:accelXZero					
A:accelYGravity			X		
A:neglectDrag					
A:pointMass					
A:freeFlight					
A:neglectCurv					
A:timeStartZero					
A:gravAccelValue					

	A:twoDMotion	**A:cartSyst**	**A:yAxisGravity**	**A:launchOrigin**	**A:ta**
DD:vecMag					
DD:speedIX					
DD:speedIY					
TM:acceleration					
TM:velocity					
GD:rectVel					
GD:rectPos					
GD:velVec	X	X			
GD:posVec	X	X			
IM:calOfLandingTime			X	X	X
IM:calOfLandingDist			X	X	
IM:offsetIM					
IM:messageIM					

	A:twoDMotion	A:cartSyst	A:yAxisGravity	A:launchOrigin	A:ta
FR:Input-Values					
FR:Verify-Input-Values					
FR:Calculate-Values					
FR:Output-Values					
NFR:Correct					
NFR:Verifiable					
NFR:Understandable					
NFR:Reusable					
NFR:Maintainable					
NFR:Portable					

	DD:vecMag	DD:speedIX	DD:speedIY	TM:acceleration	TM:velo
DD:vecMag					
DD:speedIX	X				
DD:speedIY	X				
TM:acceleration					
TM:velocity					
GD:rectVel				X	
GD:rectPos					X
GD:velVec					
GD:posVec					
IM:calOfLandingTime			X		
IM:calOfLandingDist		X			
IM:offsetIM					
IM:messageIM					

Table 9: Traceability

	DD:vecMag	DD:speedIX	DD:speedIY	TM:acceleration	TM:velo
GS:targetHit					
FR:Input-Values					
FR:Verify-Input-Values					
FR:Calculate-Values					
FR:Output-Values					
NFR:Correct					
NFR:Verifiable					

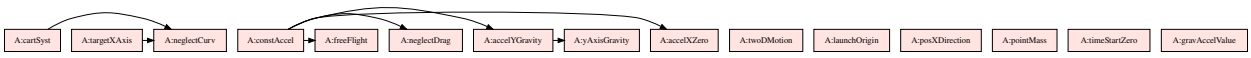

Figure 2: TraceGraphAvsA

Figure 3: TraceGraphAvsAll

	DD:vecMag	**DD:speedIX**	**DD:speedIY**	**TM:acceleration**	**TM:velo**
NFR:Understandable					
NFR:Reusable					
NFR:Maintainable					
NFR:Portable					

The purpose of the traceability graphs is also to provide easy references on what has to be additionally modified if a certain component is changed. The arrows in the graphs represent dependencies. The component at the tail of an arrow is depended on by the component at the head of that arrow. Therefore, if a component is changed, the components that it points to should also be changed. Fig:TraceGraphAvsA shows the dependencies of assumptions on the assumptions. Fig:TraceGraphAvsAll shows the dependencies of data definitions, theoretical models, general definitions, instance models, requirements, likely changes, and unlikely changes on the assumptions. Fig:TraceGraphRefvsRef shows the dependencies of data definitions, theoretical models, general definitions, and instance models with each other. Fig:TraceGraphAllvsR shows the dependencies of requirements, goal statements on the data definitions, theoretical models, general definitions, and instance models. Fig:TraceGraphAllvsAll shows the dependencies of dependencies of assumptions, models, definitions, requirements, goals, and changes with each other.

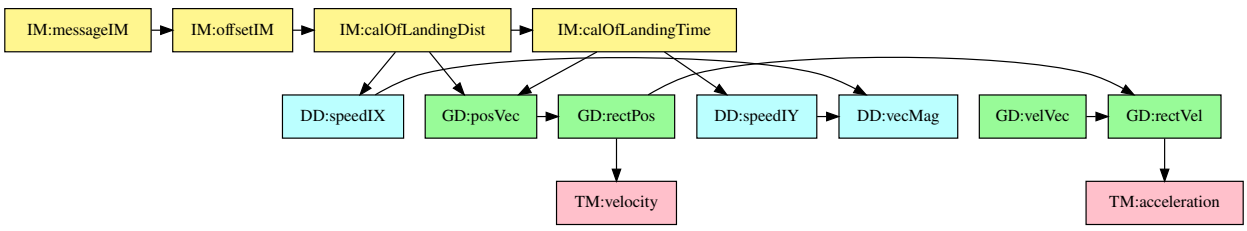

Figure 4: TraceGraphRefvsRef

Figure 5: TraceGraphAllvsR

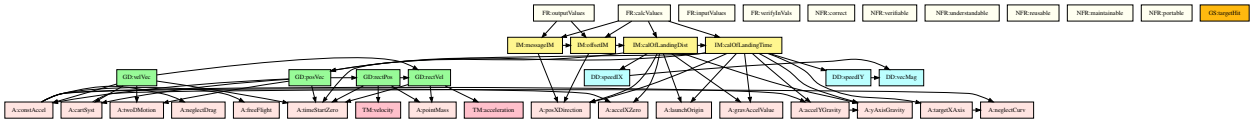

Figure 6: TraceGraphAllvsAll

For convenience, the following graphs can be found at the links below:

- TraceGraphAvsA
- TraceGraphAvsAll
- TraceGraphRefvsRef
- TraceGraphAllvsR
- TraceGraphAllvsAll

6 Values of Auxiliary Constants

This section contains the standard values that are used for calculations in Projectile.

Symbol	Description	Value	Unit
g	gravitational acceleration	9.8	$\frac{\text{m}}{\text{s}^2}$
ε	hit tolerance	2.0%	–
π	ratio of circumference to diameter for any circle	3.14159265	–

Table 11: Auxiliary Constants

7 References

[1] Wikipedia Contributors. *Acceleration*. https://en.wikipedia.org/wiki/Acceleration. June 2019.

[2] Wikipedia Contributors. *Cartesian coordinate system*. https://en.wikipedia.org/wiki/Cartesian_coordinate_system. June 2019.

[3] Wikipedia Contributors. *Velocity*. https://en.wikipedia.org/wiki/Velocity. June 2019.

[4] R. C. Hibbeler. *Engineering Mechanics: Dynamics*. Pearson Prentice Hall, 2004.