



THE UNIVERSITY *of* EDINBURGH

Edinburgh Research Explorer

AlphaImpute2: Fast and accurate pedigree and population based imputation for hundreds of thousands of individuals in livestock populations

Citation for published version:

Whalen, A, Gorjanc, G & Hickey, J 2020 'AlphaImpute2: Fast and accurate pedigree and population based imputation for hundreds of thousands of individuals in livestock populations' bioRxiv, at Cold Spring Harbor Laboratory. <https://doi.org/10.1101/2020.09.16.299677>

Digital Object Identifier (DOI):

[10.1101/2020.09.16.299677](https://doi.org/10.1101/2020.09.16.299677)

Link:

[Link to publication record in Edinburgh Research Explorer](#)

Document Version:

Publisher's PDF, also known as Version of record

General rights

Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy

The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.



1
2
3
4 **AlphaImpute2: Fast and accurate pedigree and population based imputation for hundreds**
5 **of thousands of individuals in livestock populations**

6
7 Andrew Whalen and John M Hickey*

8
9 The Roslin Institute and Royal (Dick) School of Veterinary Studies, The University of
10 Edinburgh, Midlothian, Scotland, UK

11
12 *Corresponding author

13
14 Email addresses:
15 AW: awhalen@roslin.ed.ac.uk

16 JMH: john.hickey@roslin.ed.ac.uk

17

18

19 **Declarations**

20 **Funding** The authors acknowledge the financial support from the BBSRC ISPG to The Roslin
21 Institute BB/J004235/1, from Genus PLC, and from Grant Nos. BB/M009254/1, BB/L020726/1,
22 BB/N004736/1, BB/N004728/1, BB/L020467/1, BB/N006178/1 and Medical Research Council
23 (MRC) Grant No. MR/M000370/1.

24

25 **Conflicts of interest/Competing interests**

26 On behalf of all authors, the corresponding author states that there is no conflict of interest.

27

28 **Ethics approval**

29 Not applicable

30

31 **Consent to participate**

32 Not applicable

33

34 **Consent for publication**

35 Not applicable

36

37 **Availability of data and material**

38 Not applicable

39

40 **Code availability**

41 The code used to simulate the data in this study is available from the authors upon a reasonable

42 request. The method, AlphaImpute2, is available from the authors's website
43 <https://alphagenes.roslin.ed.ac.uk/>.

44

45 **Author contributions**

46 AW designed the imputation algorithm. AW and JH designed the simulations. AW ran
47 the simulations and analyzed the results. All authors contributed to writing the manuscript and
48 approved the final manuscript

49

50 **Abstract**

51 In this paper we present a new imputation algorithm, AlphaImpute2, which performs fast
52 and accurate pedigree and population based imputation for livestock populations of hundreds of
53 thousands of individuals. Genetic imputation is a tool used in genetics to decrease the cost of
54 genotyping a population, by genotyping a small number of individuals at high-density and the
55 remaining individuals at low-density. Shared haplotype segments between the high-density and
56 low-density individuals can then be used to fill in the missing genotypes of the low-density
57 individuals. As the size of genetics datasets have grown, the computational cost of performing
58 imputation has increased, particularly in agricultural breeding programs where there might be
59 hundreds of thousands of genotyped individuals. To address this issue, we present a new
60 imputation algorithm, AlphaImpute2, which performs population imputation by using a particle
61 based approximation to the Li and Stephens which exploits the Positional Burrows Wheeler
62 Transform, and performs pedigree imputation using an approximate version of multi-locus
63 iterative peeling. We tested AlphaImpute2 on four simulated datasets designed to mimic the
64 pedigrees found in a real pig breeding program. We compared AlphaImpute2 to AlphaImpute,
65 AlphaPeel, findhap version 4, and Beagle 5.1. We found that AlphaImpute2 had the highest
66 accuracy, with an accuracy of 0.993 for low-density individuals on the pedigree with 107,000
67 individuals, compared to an accuracy of 0.942 for Beagle 5.1, 0.940 for AlphaImpute, and 0.801
68 for findhap. AlphaImpute2 was also the fastest software tested, with a runtime of 105 minutes a
69 pedigree of 107,000 individuals and 5,000 markers was 105 minutes, compared to 190 minutes
70 for Beagle 5.1, 395 minutes for findhap, and 7,859 minutes AlphaImpute. We believe that
71 AlphaImpute2 will enable fast and accurate large scale imputation for agricultural populations as
72 they scale to hundreds of thousands or millions of genotyped individuals.

73 **Introduction**

74 In this paper we present a new imputation algorithm, AlphaImpute2, which performs fast
75 and accurate pedigree and population based imputation for livestock populations of hundreds of
76 thousands of individuals. Genetic imputation is a commonly used tool in agricultural and human
77 genetics. It can be used to decrease the cost of genotyping individuals by allowing only a small
78 number of individuals to be genotyped on a high-cost high-density genotyping platform, and the
79 remaining individuals to be genotyped on a lower-cost lower-density platform. Shared haplotype
80 segments between the low-density and the high-density individuals are then used to fill in
81 missing genotypes for the low-density individuals [1,2]. Low cost genotypes are important for
82 increasing the rate of genetic gain in animal and plant breeding programs [3–5]. As genotyping
83 animals has become a routine part of breeding operations, many agricultural datasets contain
84 hundreds of thousands, or even millions of genotyped individuals [6,7] which means that
85 imputation algorithms must be to scale to ever expanding datasets.

86 Genetic imputation algorithms use either (1) pedigree or family information to perform
87 imputation, which rely on long shared haplotype segments between an individual and their
88 parents, (2) population information to perform imputation, which rely on haplotype sharing
89 between an individual and distant relatives, or (3) both sources of information in a combined
90 algorithm. Pedigree based imputation tends to be fast and accurate, but requires the pedigree of
91 the population to be known, and many of the founders to be genotyped at high density [8–11].
92 Population based imputation tend to be slower and less accurate, particularly at low marker
93 densities, but can perform imputation on individuals with unknown parents and no known
94 genotyped relatives [2,12]. Population and pedigree based imputation can be effectively
95 combined for livestock populations: pedigree information is used to impute the genotypes of

96 most individuals, and population information is used to impute the remaining genotypes,
97 particularly those of founders or individuals with ungenotyped parents [9,13,14]. When aiming
98 to improve the scaling of a combined imputation algorithm, most of the runtime tends to occur in
99 the population imputation steps [13].

100 There have been a large number improvements in the runtime of population based
101 imputation algorithms, particularly those based on the “Li and Stephens” hidden Markov model
102 framework [2]. In this framework, an individual’s genotypes are modelled as a mosaic of pairs of
103 haplotypes from the reference library. The reference library represents possible ancestral
104 haplotypes in the population, and generally consists of all of the phased haplotypes of the high-
105 density individuals. By itself, this algorithm scales poorly, with a runtime that is quadratic with
106 the number of haplotypes in the reference library. Runtime can be improved by either using a
107 fixed subset of haplotypes from the reference library [15,16], or by using a phasing algorithm to
108 pre-phase the data and running haploid hidden Markov model separately on each phased
109 chromosome [17,18]. Because the haploid hidden Markov model only needs to consider one
110 chromosome at a time, it scales linearly with the number of haplotypes in the reference panel,
111 allowing it to scale to reference panels of tens of thousands of haplotypes.

112 For reference haplotypes with hundreds of thousands of haplotypes, scaling can be
113 improved by employing the Positional Burrows Wheeler Transform (PBWT; [19]). The PBWT
114 is an opportunistic data structure which lexicographically sorts the haplotypes at each loci. By
115 sorting the library in this way, it is possible to search through the haplotype reference library for
116 a given haplotype segment in constant time (independent of the size of the library). The creation
117 of the PBWT is linear in both the number of markers and number of individuals, but once
118 created, it can be re-used for all of the individuals genotyped with the same set of markers. There

119 are a growing number of approaches for using the PBWT to speed up the runtime of imputation,
120 e.g., by using it to find a fixed-number of reference haplotypes to use for haploid imputation
121 [20], find “maximally matching” haplotype segments [19], or implement a Viterbi algorithm by
122 using a branch and bound search [21].

123 In this paper we first present a new population imputation algorithm which uses the
124 PBWT to perform a guided stochastic search through the haplotype reference library. The idea
125 behind this algorithm is to focus on combinations of haplotypes that have high posterior
126 probability. We do this by creating a series of particles and having them explore the high
127 probability paths through the haplotype reference library. Normally the number of particles we
128 would need to use would scale based on the size of the haplotype reference library. We solve this
129 issue by having the particles represent all of the haplotypes in a region with the same genotype
130 state. We then use the PBWT to update each of these particles in constant time, which allows this
131 approach to scale to large reference haplotype libraries.

132 We also present a refined version of multi-locus iterative peeling which has greatly
133 reduced runtime and memory requirements compared to previous versions [22]. Multi-locus
134 iterative peeling is a probabilistic method for performing pedigree based imputation, that has
135 high accuracy particularly in the presence of genotyping errors [9,22,23]. However, multi-locus
136 iterative peeling has traditionally been too computationally intensive to use for routine
137 imputation, and most pedigree based imputation algorithms use heuristic methods to perform
138 population imputation [8,11]. We found that it was possible to greatly increase the speed of
139 multi-locus iterative peeling by approximating the joint genotype probabilities of an individual’s
140 parents, and by calling the segregation and genotype states when estimating an offspring’s

141 contribution to their parent's genotypes. These approximations appear to have limited impact on
142 imputation accuracy.

143 Finally, we present a combined algorithm which integrates the population and pedigree
144 imputation algorithm.

145 We have implemented the population, pedigree, and combined imputation algorithms in a
146 new software package, AlphaImpute2. We compared the performance of AlphaImpute2 to
147 AlphaImpute [8], AlphaPeel [22], findhap version 4 [24], Beagle 4.1 [25], and Beagle 5.1 [26]
148 on a series of simulated datasets designed to mimic four real pig pedigrees. We find that
149 AlphaImpute2 has high accuracy and low runtimes, achieving an average accuracy of .99 for
150 low-density individuals across all four pedigrees. The runtime for imputing a single chromosome
151 of a pedigree of 107,000 individuals with 5,000 markers was just over two hours. Compared to
152 the other software, AlphaImpute2 had higher accuracy and lower run-times in most situations.

153

154 **Materials and Methods**

155 **Population imputation using particles**

156 AlphaImpute2 performs population based imputation using an approximate version of the Li and
157 Stephens algorithm [2]. In the Li and Stephens algorithm models an individual's genotype is
158 constructed as a mosaic of haplotypes from a haplotype reference panel. This can be
159 implemented in a hidden Markov model where the state space consists of pairs of haplotype
160 identifiers from the reference panel at each loci, and inference is done to find a high-likelihood
161 path, or sequence of haplotype identifiers, through this space. The path can then be used to
162 impute and phase missing genotypes by looking at that state of each haplotype along the path.

163 Because the state space grows quadratically with the size of the reference panel, approximations
164 are needed to perform inference.

165 Lunter [21] published an algorithm to produce an exact maximum likelihood (Viterbi)
166 path for a diploid Li and Stephens algorithm in constant time. This approach uses the
167 Positional Burrows Wheeler Transform [19] as an opportunistic data structure that allows
168 searching across many similar haplotypes at the same time. The approach we use here is loosely
169 based on the framework of Lunter [21]. Instead of taking the maximum likelihood path, we
170 instead generate samples from an approximate posterior distribution over all possible paths
171 through the haplotype reference library. The logic behind this choice is that there may be many
172 paths with similarly high likelihoods, and by combining information from multiple samples it
173 may be possible to obtain more accurate genotypes than from any single sample.

174 To run the algorithm we construct a series of series of a particles. Each particle consists
175 of a pair of ranges of haplotypes at each loci, (ϕ_x^i, ψ_y^i) where ϕ_x^i and ψ_y^i give the set of
176 haplotypes at loci i , whose values at preceding loci are given by the sequences
177 $x = \{x_{i-n}, x_{i-n+1}, \dots, x_i\}$ or $y = \{y_{i-n}, y_{i-n+1}, \dots, y_i\}$. ϕ_x^i gives the haplotypes for the paternal
178 chromosome, and ψ_y^i gives the haplotypes for the maternal chromosome. At each loci we
179 probabilistically update the each particle to a new state $(\phi_{x^{**}}^{i+1}, \psi_{y^{**}}^{i+1})$ using a guided stochastic
180 search algorithm.

181 To create the final imputed genotypes and haplotypes, we use this approach to generate
182 between 40-100 particles, and then merge the particles to form a set of consensus haplotypes.
183 These haplotypes can be used to either phase high-density individuals, or impute low-density
184 individuals.

185

186 *Updating a particle*

187 At each loci, particles are updated by probabilistically selecting a new genotype state, and
 188 recombination state. We consider a 4x4 grid of possible steps that the particle can take. In the
 189 first dimension the particle can move to one of the four phased genotype states. In the second
 190 dimension the particle can make the move by having a recombination on either none of their
 191 haplotypes, on either their maternal, or paternal haplotypes, or on both haplotypes. An example
 192 of this update is given in Figure 1.

193 The probability of selecting to move to genotype g_{i+1} with recombination r_i is:

$$p(g_{i+1}, r_i | \phi_x^i, \psi_y^i) \propto f(g_{i+1}^p | r_{i+1}^p, \phi_x^i) f(g_{i+1}^m | r_{i+1}^m, \psi_x^i) p(r_i^p) p(r_i^m) p(g_i | d_i) \#(1)$$

194 Where $f(g_i^p | r_i^p, \phi_x^i)$ is a function which gives the probability of the paternal part of the genotype
 195 state given the recombination state (for either the paternal or maternal haplotypes) and the
 196 current set of haplotypes considered; $p(r_i^p)$ is the probability of the paternal part of the
 197 recombination state; and $p(g_i | d_i)$ is the probability of the full genotype state conditional on the
 198 data observed; $p(g_i | d_i)$ is the same as in multi-locus iterative peeling and is given later, in
 199 Equation 8.

200 We calculate $f(g_i^p | r_i^p, \phi_x^i)$ as

$$f(g_{i+1}^p = 0 | r_{i+1}^p, \phi_x^i) = \begin{cases} \frac{|\phi_{(x,0)}^{i+1}|}{|\phi_{(x,0)}^{i+1}| + |\phi_{(x,1)}^{i+1}|} & \text{if } r_{i+1}^p \text{ does not indicate a recombination} \\ \frac{|\phi_0^{i+1}|}{|\phi_{(0)}^{i+1}| + |\phi_{(1)}^{i+1}|} & \text{if } r_{i+1}^p \text{ indicates a recombination} \end{cases} \#(2)$$

201 where $|\phi_{(x,0)}^{i+1}|$ gives the number of haplotypes in ϕ_x^i that have a 0 at locus $i + 1$. Similarly
 202 $|\phi_{(x,1)}^{i+1}|$ gives the number of haplotypes in ϕ_x^i that have a 1 at locus $i + 1$. ϕ_0^{i+1} represents the set
 203 of all haplotypes that have a 0 at locus $i + 1$. Using the Positional Burrows Wheeler Transform

204 we can calculate $|\phi_{x,0}^{i+1}|$ in constant time, independent of the size of the haplotype reference
205 library or the number of haplotypes in ϕ_x^i [21].

206 We calculate the probability of the maternal and paternal recombination $p(r_i^p)$ as either:

$$p(r_i^p) = \begin{cases} 1 - \gamma & \text{if } r_{i+1}^p \text{ does not indicate a recombination} \\ \gamma & \text{if } r_{i+1}^p \text{ indicates a recombination} \end{cases} \#(3)$$

207 Where γ is a recombination rate which, we estimate as $\frac{1}{n_{loci}}$. This value underestimates of the
208 effective recombination rate since individuals are inheriting haplotypes from distant relatives
209 [16], however the accuracy of this algorithm seems to largely insensitive to the recombination
210 rate given, and this serves as a good approximation.

211 In order to determine how well a particle matches the data at a particular locus, we assign
212 a score to each particle at each locus:

$$score_i = \log \left(p(r_i^p) p(r_i^m) p(d_i | g_i) \right) \#(4)$$

213 , which gives a higher score to particles that do not have a recombination and fit the observed
214 genotype data. We use the score to combine particles into a single consensus haplotype.

215

216 *Combining particles*

217 We create a consensus haplotype and genotype state out of multiple particles by using a two-
218 stage approach. In the first stage we call consensus genotypes for each locus. In the second stage
219 we phase the loci that are called as heterozygous.

220 To create a consensus genotype, we score each particle at each locus by taking the sum of
221 the scores of each particle (given by Equation 4) within a 50 marker window. We then select the
222 genotype state of the particle with the best score as the called genotype. If multiple particles have
223 the highest score, we use the most frequent genotype state of those particles the called genotype.

224 In the second stage we phase heterozygous loci by looking at transitions between
225 neighbouring heterozygous states. We track whether the alternative alleles are on the same or
226 different haplotypes, e.g., whether the phased genotype state transitions from aA to aA where the
227 alternative allele is on the maternal haplotype, or from aA to Aa where the alternative allele
228 transitions between the maternal to the paternal haplotypes. We pick the transition that is most
229 frequent in all of the particles that are heterozygous at both loci.

230

231 *Backward information*

232 In a traditional hidden Markov model, inference is done by combining information from a
233 forward pass (information from the first locus to the current locus) with a backward pass
234 (information from the last locus to the current locus). The search algorithm we present only takes
235 into account information in the forward pass, and does not make selections based on genotype
236 data from loci after the current locus. Information from backward pass can be useful for
237 imputation by filling in spontaneous missing markers, and phasing genotype states. To
238 incorporate backwards information, we first run a series of particles in reverse, i.e., from the end
239 of the chromosome to the beginning of the chromosome, and then run a forward pass of particles
240 where we replace $p(g_i|d_i)$ in Equation 1 with:

$$p(g_i|d_{i:n}, \phi, \psi) = p(g_i|d_i) \sum_{r_i} f(g_i^p|r_i^p, \phi_x^{i-1}) f(g_i^m|r_i^m, \psi_x^{i-1}) p(r_i^p) p(r_i^m) \#(5)$$

241 In this equation we project each of the reverse particles forward by one locus to see what
242 genotype state they are likely to carry at the next locus. When multiple particles are run on the
243 backward pass, $p(g_i|d_{i:n}, \phi, \psi)$ is averaged across all particles in the backward pass.

244

245 *Imputation*

246 This algorithm can also be used to impute missing markers. To perform imputation, we evaluate
247 particles on the non-missing markers. We track the loci where recombination occurs, and what
248 the set of haplotypes are for each particle at those loci. This creates an ordered pair of haplotype
249 regions $\{(start, stop), (\phi_{x^*}^{stop}, \psi_{y^*}^{stop})\}$. For each locus, we select the interval that contains the
250 locus, and fill in missing markers on the corresponding haplotypes from the middle haplotype in
251 $\phi_{x^*}^{stop}$ on the paternal side, and $\psi_{y^*}^{stop}$ on the maternal side.

252

253 *Creating the haplotype library*

254 In many animal populations pre-phased haplotype libraries are not available and so the haplotype
255 used for population imputation needs to be constructed. This is done by iteratively (1)
256 constructing a haplotype library from the high-density individuals, (2) phasing those individuals
257 with that haplotype library, and (3) re-building the library using the phased haplotypes from the
258 previous iteration. To initialize the haplotype library, we randomly phase heterozygous loci and
259 fill in spontaneous missing genotypes. We then run 5 rounds of phasing and imputation,
260 rebuilding the haplotype library at the end of each round. Running more rounds of phasing and
261 imputation can increase the quality of the haplotype library, but we found that 5 rounds was
262 sufficient in pilot simulations for accurate imputation.

263 During this process we keep track of location of which haplotypes the individual
264 contributes to the haplotype library, and remove those haplotypes as options for the particle
265 steps. This is done by modifying $|\phi_{x,0}^{i+1}|$ to be either

$$|\phi_{x,0}^{i+1}| = \begin{cases} |\phi_{x,0}^{i+1}| & \text{if } \phi_{x,0}^{i+1} \text{ contains none of the individual's haplotypes} \\ |\phi_{x,0}^{i+1}| - 1 & \text{if } \phi_{x,0}^{i+1} \text{ contains one of the individual's haplotypes} \\ |\phi_{x,0}^{i+1}| - 2 & \text{if } \phi_{x,0}^{i+1} \text{ contains both of the individual's haplotypes} \end{cases} \quad \#(6)$$

266

267 *Array Clustering*

268 In order to run this algorithm, we want $p(g_i|d_i)$ to be as informative as possible for each locus.

269 Having an informative value for $p(g_i|d_i)$ allows us to have more confidence in taking each step.

270 If the individual does not have genotype data at a locus, then $p(g_i|d_i)$ will be relatively

271 uninformative at that locus. One solution would be to only evaluate the individual's genotypes at

272 non-missing loci. This would require re-calculating the Positional Burrows Wheeler Transform

273 on an individual-by-individual basis to take into account each individual's pattern of missing

274 genotypes, which would be prohibitively computationally expensive. Instead we cluster

275 individuals based on the SNP array they are genotyped on, and build the Positional Burrows

276 Wheeler Transform on an array-by-array basis. This greatly reduces the number of times the

277 Positional Burrows Wheeler Transform needs to be calculated, and backwards information is

278 used to guide decisions on any remaining spontaneous missing markers.

279

280 **Pedigree based imputation using approximate multi-locus iterative peeling**

281 AlphaImpute2 performs pedigree based imputation using an approximate version of multi-locus

282 iterative peeling [9,22]. In an iterative peeling framework the probability of an individual's

283 genotypes is based on three sources of information [23]:

$$p(g_i) = \text{anterior}(g_i)\text{posterior}(g_i)\text{penetrance}(g_i) \#(7)$$

284 , where the anterior term represents information about an individual's genotypes based on

285 information from the ancestors of the individual filtered through their parents, the posterior

286 represents information about an individual's genotypes based on the decedents of the individual

287 filtered through their offspring, and the penetrance term represents information about an
288 individual's genotypes based on their own genetic data (i.e., SNP array or sequence data).

289 In order to take linkage information into account, multi-locus iterative peeling builds on
290 the normal iterative peeling framework by having the anterior terms and the posterior terms
291 depend on the segregation state of the individual, i.e., which pair of parental haplotypes that
292 individual inherited at each loci [22].

293 Performing exact inference in a peeling framework is challenging because the anterior
294 and posterior terms for an individual, depend on the genotypes of their parents and offspring,
295 which themselves need to be estimated. Because of this, AlphaImpute2 takes an iterative
296 approach to update the anterior and posterior terms in a series of passes up and down the
297 pedigree. This is summarized in the following algorithm:

- 298 1. Downward pass: Starting from individuals in the first generation to the last generation
 - 299 a. Re-estimate the segregation probabilities for each individual.
 - 300 b. Re-estimate the anterior terms for each individual based on their parent's
301 genotypes.
- 302 2. Upward pass: starting from the last generation to the first generation
 - 303 a. Re-estimate the segregation probabilities for each individual.
 - 304 b. Re-estimate the posterior terms for each individual based on their offspring's
305 genotypes.

306 In order to enable these passes, we sort the pedigree according to an individual's generation. The
307 generation of each individual is the minimum of the generation of their sire and dam plus one.

308 The peeling algorithm needs to be run in a series of passes. We have found that 5 passes
309 of peeling is often enough to obtain high-quality genotype probabilities. The purpose of running

310 multiple passes is primarily to transmit information horizontally across the pedigree, i.e.,
311 between children of a shared parent. The amount of information that is passed horizontal quickly
312 decays as individuals become more genetically distant.

313 In order to perform peeling we need to specify how we calculate the penetrance term, and
314 how we update the anterior, posterior, and segregation probabilities.

315

316 *Calculating the penetrance term*

317 The penetrance term give the probability of the observed genetic data, conditional on the
318 individual's genotype state. We consider four phased genotype states, aa, aA, Aa, AA, where the
319 first allele is the paternal allele, and the second allele is the maternal allele. We assume that the
320 observed genotype data is the number of observed alternative alleles for SNP data.

$$penetrance(g_i) = p(d_i|g_i) = \begin{cases} 1 - e & \text{if } g_i \text{ is consistent with } d_i \\ e & \text{otherwise} \end{cases} \quad \# \quad (8)$$

321 Where e represents the genotyping error rate with a default value of 0.01.

322

323 *Updating the anterior term*

324 In each downward pass the anterior term is updated for each individual. To perform this update
325 we re-estimate which genotypes the individual inherited from their parents using the current
326 estimate of their parents' genotypes. We calculate the anterior term for individual o at locus i as:

$$anterior(g_{o,i}) = \sum_{seg_{o,i}} \sum_{g_{s,i}, g_{d,i}} p(g_{o,i}|g_{s,i}, g_{d,i}, seg_{o,i}) p^*(g_{s,i}) p^*(g_{d,i}) p^*(seg_{o,i}) \quad \#(9)$$

327 where $p(g_{o,i}|g_{s,i}, g_{d,i}, seg_{o,i})$ is a transmission function which gives the probability that the
328 offspring inherits a genotype conditional on the offspring's segregation states and the genotypes
329 of their sire and dam. This value will be either 0 or 1 depending on if an inherited genotype is

330 consistent with the individual's segregation state and the genotypes of their parents. The term
 331 $p^*(g_{s,i})$ gives the probability of the genotype state of the sire, based on the information from the
 332 last pass of peeling.

333 This formulation of the anterior term is an approximation to the term used in previous
 334 papers [22,23]. In the traditional peeling framework the parents genotypes are given by
 335 $p_{-o}(g_{s,i}, g_{d,i})$ which gives the joint genotype probabilities of the parents ignoring information
 336 from the offspring, o . By not excluding the offspring's genotypes we are effectively double
 337 counting the offspring's genotypes: first information on the offspring's genotypes is used in the
 338 penetrance function, and second information from the offspring's genotypes will be used to
 339 calculate the posterior term of the parent, which will then be used to estimate the anterior term of
 340 the offspring. In practice, we find that the posterior term from a single offspring only provides a
 341 small amount of information to their parent, and that the double counting of information here
 342 does not lead to a substantial loss in accuracy. We also assume that the genotype probabilities of
 343 the parents are independent, i.e., $p(g_{s,i}, g_{d,i}) = p(g_{s,i})p(g_{d,i})$.

344

345 *Updating the posterior term*

346 In each upward pass the posterior term is updated based on the genotypes of the offspring. This
 347 update is performed on a family-by-family basis and the result is combined across families. For a
 348 sire, s , with mates, $M = \{m_1, m_2, \dots\}$ the posterior term is:

$$posterior(g_{s,i}) \propto \prod_{m \in M} posterior_m(g_{s,i}) \quad \#(10)$$

349 where posterior term for each mate is given by

$$posterior_m(g_{s,i}) \propto \sum_{g_{m,i}} \prod_o \sum_{seg_{o,i}} \sum_{g_{o,i}} p(g_{o,i} | g_{s,i}, g_{d,i}, seg_{o,i}) p_{-parents}^*(g_{o,i}) p^*(seg_{o,i}) p^*(g_{o,i}) \quad \#(11)$$

350

351 Where the product is on all of the shared offspring between m and s . Similar terms are used to
352 generate the posterior estimate for each dam.

353 In order to avoid double counting the genotypes of the parents, we exclude the anterior
354 term from the calculation of the genotypes of the offspring in $p_{-parents}(g_{o,i})$. As when
355 calculating the anterior term, we do not exclude the contribution of the offspring when
356 calculating the genotypes of the mate $p(g_{m,i})$.

357

358 *Calculating the probability of each segregation state*

359 The probability of each segregation state are calculated by using a hidden Markov model
360 to determine which segregation state the individual carries at different loci across the genome.
361 We consider four segregation states (mm, pm, mp, pp) where the first letter gives whether the
362 individual inherits their sire's maternal or paternal haplotype, and the second letter gives whether
363 the individual inherits their dam's maternal or paternal haplotype.

364 Hidden Markov models are defined by a series of emission probabilities which give the
365 likelihood of the observations given a hidden state, and transmission probabilities, which give
366 the probability of transitioning between hidden states. The emission probabilities of this model
367 are:

$$p(seg_{o,i}) \propto \sum_{g_{o,i}} \sum_{g_{d,i}, g_{s,i}} p(g_{o,i} | g_{s,i}, g_{d,i}, seg_i) p^*(g_{s,i}) p^*(g_{d,i}) p_{-parents}^*(g_o) \#(12)$$

368 Which uses Bayes' rule to estimate the probability of each segregation state conditional on the
369 estimated genotypes of the individual and their parents. The transmission function is given by
370 (Whalen 2018):

$$p(seg_i = s | seg_i = s') = (1 - \gamma)^{2-d} \gamma^d \quad \#(13)$$

371 where d is the number of recombinations required to move between s and s' , i.e.,

372 $p(seg_{i,j} = pp | seg_{i,j-1} = pm) = (1 - \gamma)\gamma$, and γ is the recombination rate. We found that

373 accuracy was largely insensitive to the recombination rate and so set it a default value of $\frac{1}{n_{loci}}$

374 where n_{loci} is the number of loci on the chromosome. This assumes markers are evenly spaced

375 (in genetic map distance) across an 100 cM chromosome. The assumed total chromosome

376 genetic map length can be changed using a command line option.

377 We use the forward-backward algorithm [27] to calculate segregation probabilities across

378 each loci. To simplify the amount of information stored at each loci, we assume the segregation

379 probabilities for the maternal and paternal haplotypes are independent and set e.g.,

$$380 p(seg_{o,pat,i} = m) = p(seg_{o,i} = mp) + p(seg_{o,i} = mm).$$

381

382 *Calling genotype probabilities for the posterior term*

383 In order to reduce runtime we call the offspring segregation and genotype probability values

384 when calculating the posterior terms for their parents. We use calling threshold of 0.99 for

385 calling the segregation values, and a calling threshold of 0.99 for the genotypes on the first round

386 of peeling, and a threshold of 0.95 for subsequent rounds of peeling. For genotype or segregation

387 probabilities that do not reach the threshold, the genotypes or segregation values are set with

388 each state being equally likely.

389 By calling the segregation values and genotype values, we are able to store part of the

390 posterior update,

$$\sum_{seg_{o,i}} \sum_{g_{o,i}} p(g_{o,i} | g_{s,i}, g_{d,i}, seg_i) p_{-parents}(g_{o,i}) p(seg_{o,i}) \quad \#(14)$$

391 , in a look-up table, which substantially reduces runtime. In addition, we do not consider the
392 dependency between the uncertainty of segregation states at nearby loci. The un-modelled linked
393 uncertainty between segregation states can lead to errors in imputing the parental genotypes. By
394 calling the segregation probabilities we mitigate the impact of this simplification by only
395 considering non-equal segregation probabilities where there is minimal uncertainty in the
396 segregation state.

397 After running the final round of peeling we also call the genotype probabilities of all
398 individuals in the population to get best-guess genotypes for each individual.

399

400 **Integrating population and pedigree based imputation**

401 Past work has found that combining pedigree and population imputation algorithms can increase
402 accuracy in populations where pedigree information is available [13,14]. The goal of this
403 combination is to use the population-based imputation algorithms to phase and impute the
404 individuals at the top of the pedigree. These genotypes can then be dropped through the rest of
405 the pedigree using the pedigree based imputation algorithm. To combine the pedigree and
406 population algorithms in AlphaImpute2, we perform imputation using a three step approach
407 where we first perform an initial run of pedigree imputation, we then perform population
408 imputation on a limited set of “pseudo-founders”, and we finish with a final run of pedigree
409 imputation to fill in the remaining missing genotypes.

410

411 *Step 1: Initial pedigree imputation*

412 In Step 1, 5 rounds of multi-locus iterative peeling are run on the population. After the final
413 round all of the genotypes in the population are called with a genotype calling threshold of 0.9.

414 We then split the population to three parts: (1) high-density individuals that have fewer than 10%
415 missing markers, (2) low-density individuals who are “pseudo-founders” (see below) and (3)
416 low-density individuals who are not “pseudo founders”. After splitting the population, the
417 genotypes of individuals in group (3) are reset to their original genotype values before pedigree
418 based imputation.

419 Pseudo-founders are individuals who genotyped at a higher density than their parents
420 (accounting for the fact that their parents may be imputed to a higher density using pedigree
421 based imputation). To detect pseudo-founders we go through the pedigree from the start to the
422 end and calculate the effective genotyping density of an individual:

$$score_{ind} = \begin{cases} \min(score_{sire}, score_{dam}) & (a) \text{ if } missing_{ind} * 0.9 < \min(score_{sire}, score_{dam}) \\ missing_{ind} & (b) \text{ otherwise} \end{cases}$$

423 where $missing_{ind}$ is the percentage of non-missing markers the individual has. The value 0.9 is
424 used to give a slight preference to using the genotype of the parents if the individuals are at a
425 similar marker density. Individuals in group (b) are the “pseudo-founders” of the population.

426

427 *Step 2: Population imputation*

428 In Step 2, we use the population imputation algorithm to phase the high-density individuals
429 detected in Step 1, and use the haplotype constructed from their phased haplotypes as the
430 reference library to impute the low-density “pseudo-founders”. We perform an initial 5 rounds of
431 phasing to iteratively build the reference haplotype library using 40 particles to phase each
432 individual. For imputation we use 100 particles to impute each individual. At the end of this step,
433 we reset the genotypes and haplotypes of high-density individuals that are not “pseudo-founders”
434 to their original genotype states at the start of Step 1. The number of particles selected for

435 phasing and imputation were chosen based on pilot simulations. Larger numbers of particles may
436 yield more accurate results, but the improvement in accuracy will likely be small.

437

438 *Step 3: Final pedigree imputation*

439 In Step 3, we re-run 5 rounds of multi-locus iterative peeling, using the new phased genotypes
440 for the “pseudo-founder” generated in Step 2. In order to reduce the negative impact of switch
441 errors, we perform peeling on a lesioned pedigree where the link between a “pseudo-founder”
442 and both of their parents is removed. After running multi-locus iterative peeling, the genotypes
443 are set to the best-guess genotypes.

444

445 **Testing the algorithm**

446 We tested the performance of AlphaImpute2 on four simulated datasets based on pedigrees taken
447 from a commercial pig breeding program. The pedigrees had either 18,349 (*18k*), 34,425 (*34k*),
448 63,872 (*63k*), or 107,815 (*107k*) individuals and were genotyped on four SNP arrays which
449 ranged from 350 (*very low density*), 10,000 (*low density*), 33000 (*medium density*), and 46,000
450 (*high density*) markers. Although these marker densities are lower than highest density SNP
451 arrays available for humans and livestock, they represent commonly used marker densities for
452 performing genomic selection in many animal breeding programs [28–30].

453 We compared the performance of AlphaImpute2 to that of Beagle 4.1, Beagle 5.1,
454 AlphaImpute, AlphaPeel, and findhap. We evaluated each software on their accuracy, runtime,
455 and memory requirements.

456

457

458 **Simulated genetic data**

459 We simulated the four pedigrees by generating a set of founder haplotypes using a Markovian
460 Coalescent Simulator [31], and then dropped them through each pedigrees.

461 The founder haplotypes were generated by assuming there were 18 100-cM long
462 chromosomes that were simulated using a per site mutation rate of 2.5×10^{-8} , and an
463 effective population size (N_e) that changed over time based on estimates for the Holstein cattle
464 population [32]. N_e was set to 100 in the final generation of simulation and to 1256, 4350, and
465 43,500 at 1000, 10,000, and 100000 generations ago, with linear changes in between. The
466 number of markers per chromosome varied between 1,231 and 4690 based on the marker
467 densities on each chromosome in the real genotype data.

468 The founder haplotypes were then dropped through the pedigree using AlphaSimR [33].
469 The genotypes of each individual were then masked to reflect the pattern of missingness for that
470 individual in the real genotype data.

471

472 **Comparison with other software**

473 We evaluated the performance of AlphaImpute2 when using either the population only
474 algorithm, the pedigree only algorithm, or the combined algorithm.

475 We compared the performance of AlphaImpute2 with the performance Beagle 4.1,
476 Beagle 5.1, AlphaPeel, findhap, and AlphaImpute. Beagle 4.1 and Beagle 5.1 were run using
477 default parameters except the effective population size which was set to 200. AlphaImpute and
478 AlphaPeel were run with default parameters. For AlphaImpute we rounded the genotype
479 probabilities that it outputs before calculating accuracy to make it consistent with the other

480 software packages. findhap was run with the recommend parameters of maxlen = 600, minlen =
481 75, and errate = .004.

482 Our goal in running a large number of other software packages was to evaluate the
483 performance of both the population only, and pedigree only algorithms separately, and to
484 evaluate the performance of the combined algorithm.

485 Beagle 4.1 and Beagle 5.1 were chosen to serve as a benchmark for the population only
486 algorithm. Both software packages are commonly used in the human and animal imputation
487 literature, and Beagle 5.1 has incorporated a number of (as of yet unpublished) improvements for
488 phasing.

489 AlphaPeel was chosen to serve as a benchmark for the pedigree only algorithm.
490 AlphaPeel implements a version of multi-locus iterative peeling, which is approximated by the
491 pedigree only algorithm in AlphaImpute2. Our goal in making this comparison was to see how
492 much accuracy was sacrifice to increase runtime in AlphaImpute2.

493 findhap and AlphaImpute were chosen to serve as a benchmark for a combined pedigree
494 and population imputation algorithm. Both programs are currently in use in commercial breeding
495 programs, and AlphaImpute2 could serve as a possible candidate to replace them.

496

497 **Performance measurements**

498 Imputation accuracy was measured as the correlation between an individual's imputed
499 genotype and their true genotype, corrected for the population minor allele frequency [34]:

$$accuracy = cor(G_{impute} - 2 maf, G_{true} - 2 maf)$$

500 Accuracy was averaged across all of the 18 simulated chromosomes.

501 We also measured the runtime and memory usage of each program. All programs were
502 run on the Edinburgh Compute and Data Facility cluster using 4 cores. Programs were given at
503 most eight days to impute each chromosome. The results are given only for programs that
504 successfully finished on all of the chromosomes.

505

506 **Results**

507 We found that AlphaImpute2 had high accuracy and low run-times across all four pedigrees.
508 Imputation accuracy for the 107k pedigree was .988 for high-density individuals, .988 for
509 medium density individuals, .993 for low-density individuals, and .81 for very-low-density
510 individuals. Imputation took 105 minutes and 14.4 GB of memory for Chromosome 1 (4,600
511 Markers and 107,000 individuals). AlphaImpute2 had higher accuracy than the alternative
512 algorithms and comparable run times to findhap and Beagle 5.1, both of which are significantly
513 faster than Beagle 4.1. The accuracy, runtime, and memory usage of each algorithm is given in
514 Table 1.

515

516 **Accuracy of the full AlphaImpute2 algorithm**

517 The accuracy of AlphaImpute2 depended primarily on the genotyping density of the
518 individuals and their relative position in the pedigree. We found similar accuracies across all four
519 pedigrees and so focus on the 18k pedigree to enable comparisons to Beagle 4.1 which did not
520 finish on all pedigrees.

521 On the 18k pedigree the accuracy of the full AlphaImpute2 algorithm was .998 for high-
522 density individuals, .944 for medium-density individuals, .990 for low-density individuals, and
523 .827 for very-low-density individuals. The lower accuracy for medium-density individuals

524 compared to low-density individuals was likely driven by their relative position in the pedigree.
525 All of the medium-density individuals appeared in the first quarter of the pedigree compared to
526 only 3% of the low-density individuals and 0.2% of the high-density individuals.

527 The accuracy of the pedigree only algorithm was .998 for high-density individuals, .661
528 for medium-density individuals, .987 for the low-density individuals, and .862 for the very-low-
529 density individuals. Compared to the full algorithm, the pedigree only algorithm had much lower
530 accuracy on the medium-density individuals (.661 compared to .944), and similar accuracies on
531 the high-density, low-density, and very-low-density individuals.

532 The accuracy of the population only algorithm was .987 for the high-density individuals,
533 .929 for the medium-density individuals, .973 for the low-density individuals, and .257 for the
534 very-low-density individuals. Compared to the full algorithm, the population only algorithm had
535 much lower accuracy on the very-low-density individuals (0.257 compared to 0.827), and
536 between 1-2% lower accuracies on the high-density, medium-density, and low-density
537 individuals.

538 The full AlphaImpute2 algorithm had higher accuracy than both the population only or
539 pedigree only algorithms except in the case of very-low-density individuals where the pedigree
540 only algorithm had a slightly higher accuracy (.862 compared to .827).

541

542 **Pedigree only imputation accuracy compared to AlphaPeel**

543 The pedigree only algorithm in AlphaImpute2 uses an approximate version of multi-locus
544 iterative peeling that is implemented in AlphaPeel. The accuracy of the two algorithms are
545 similar, with the accuracy of AlphaPeel on the 18k pedigree being .997 for high-density

546 individuals, .733 for medium-density individuals, .984 for low-density individuals, .855 for very-
547 low-density individuals.

548 The correlation between the genotypes imputed by the pedigree-only imputation
549 algorithm and AlphaPeel was high. On Chromosome 1 for the 18k pedigree, the correlation
550 between the genotypes imputed between the two algorithms was on average .973, with a
551 correlation of .999 for high-density individuals, .960 for medium-density individuals, .994 for
552 low-density individuals, and .804 for very-low-density individuals. The lower correlation for the
553 medium-density and very-low-density individuals is due to the lack of high-density parents for
554 these individuals. In AlphaPeel, the observed minor allele frequency is used as a prior for the
555 genotypes of the founder individuals, whereas a minor allele frequency of 0.5 is used as a prior
556 for founder individuals in AlphaImpute2. We return to this difference in the Discussion.

557

558 **AlphaImpute2 accuracy compared to Beagle 4.1 and Beagle 5.1**

559 For the 18k pedigree, the accuracy of Beagle 4.1 was .995 for the high-density
560 individuals, .944 for the medium-density individuals, .969 for the low-density individuals, and
561 .327 for the very-low-density individuals. The accuracy of Beagle 5.1 was .626 for the high-
562 density individuals, .909 for the medium-density individuals, .939 for the low-density
563 individuals, and .219 for the very-low-density individuals.

564 The accuracy of Beagle 4.1 was slightly higher than that of Beagle 5.1 in all cases, with
565 the largest difference being on filling spontaneous missing markers in the high-density
566 individuals, where the accuracy of Beagle 4.1 was .995 but the accuracy of Beagle 5.1 was .626.

567 The accuracy of the population only algorithm in AlphaImpute2 was similar to Beagle
568 4.1 with lower accuracies on the medium-density individuals (.929 compared to .944), and very-
569 low-density individuals (.257 compared to .327).

570

571 **Combined algorithms: findhap and AlphaImpute2**

572 For the 18k pedigree, the accuracy of findhap was .719 for the high-density individuals,
573 .627 for the medium-density individuals, .774 for the low-density individuals, and .445 for the
574 very-low-density individuals. The accuracy of findhap was between 20-40% lower than
575 combined algorithm in AlphaImpute2 in all cases (Table 1).

576 The accuracy of AlphaImpute2 was .940 for the high-density individuals, .875 for the
577 medium-density individuals, .931 for the low-density individuals, and .641 for the very-low-
578 density individuals. The accuracy of the combined algorithm in AlphaImpute2 was higher than
579 AlphaImpute in all cases, with the largest differences for medium density individuals (.944
580 compared to .857) and very-low-density individuals (.827 compared with .641).

581

582 **Runtime**

583 AlphaImpute2 was faster than all of the other software packages tested. For the pedigree
584 of 18k individuals, AlphaImpute2 took 15 minutes, followed by findhap which took 17 minutes,
585 Beagle 5.1 which took 28.1 minutes, AlphaImpute which took 348 minutes, and Beagle 4.1
586 which took 2,250 minutes.

587 For the pedigree of 107k individuals, AlphaImpute2 took 105 minutes, Beagle 5.1 took
588 190 minutes, findhap took 395 minutes, and AlphaImpute took 7,859. Beagle 4.1 did not finish
589 on the 107k pedigree within eight days of run time.

590

591 **Discussion**

592 In this paper we present a new population and pedigree based imputation algorithm,
593 AlphaImpute2, and demonstrate its performance on four simulated datasets based on real
594 livestock pedigrees. We find that it is able to perform fast and accurate imputation in a range of
595 scenarios, and preforms competitively with other existing imputation software. In the remainder
596 of the discussion we discuss the advantages of combining pedigree and population based
597 imputation information for imputation, ways to further decrease the runtime of AlphaImpute2,
598 the performance of the approximate iterative peeling framework, compare the population
599 imputation algorithm to already existing population imputation algorithms, and the particle based
600 approach for approximating the Li and Stephens algorithm.

601

602 **Combining pedigree and population imputation increases accuracy**

603 In line with previous research, we find that combining population and pedigree
604 imputation can increase accuracy compared to running either the population or pedigree
605 imputation algorithms alone [13].

606 Compared to the pedigree only algorithm, the combined algorithm delivers high-
607 accuracy phasing and imputation for the individuals at the top of the pedigree. These phased
608 genotypes can then be used to impute individuals further down in the pedigree. This improves
609 the imputation accuracy for both the “pseudo-founders” of the pedigree, but also other
610 descendants who may be genotyped at lower densities. A similar effect was seen in LDMIP
611 which used a population based imputation algorithm to impute and phase the founders of the
612 pedigree before running multi-locus iterative peeling [9].

613 Compared to the population only algorithm, the combined algorithm delivers higher-
614 accuracy imputation across the board, particularly for very-low-density individuals. For these
615 individuals imputation accuracy is improved by using pedigree information to decrease the
616 number of haplotypes that need to be considered – the four parental haplotypes in the case of
617 pedigree based imputation, compared to tens of thousands for population imputation – which
618 makes it easier to find the correct haplotypes with a limited number of low-density markers.

619 The only place where accuracy of the combined algorithm was lower than that of the
620 pedigree only algorithm, was for very-low-density individuals, particularly those at the
621 beginning of the pedigree. The lower accuracy on very-low-density individuals is likely due to a
622 lack of high-density or medium-density ancestors for these individuals. In AlphaPeel, the minor
623 allele frequency is used as a prior for missing genotypes of founders in the population. This
624 allows AlphaPeel to take the uncertainty in the genotypes of these individuals into account. In
625 contrast, in the combined algorithm the founders and “pseudo founders” are imputed with the
626 population imputation algorithm, and the resulting genotypes are treated as observed genotypes
627 (with a default 1% error rate). The population imputation algorithm tends to have low error rates
628 for high, medium, and low-density individuals, but high error rates for very-low-density
629 individuals. Treating these imputed genotypes as observed in the final round of population
630 imputation may be the cause of the lower imputation accuracy. A solution to this problem may
631 be to include a minimum genotyping density required for population imputation (e.g., 10-50 non-
632 missing markers per chromosome), and using the minor allele frequency as a prior for the
633 genotypes of “pseudo founders” who do not reach this density.

634

635 **Decreasing the runtime of AlphaImpute2**

636 We found that the combined imputation algorithm had lower runtime than the population
637 only algorithm, but a higher runtime than the pedigree-only algorithm. The lower runtime
638 compared to the population only algorithm is likely due to the fact that in the combined
639 algorithm imputation is only run on a small set of “pseudo founders”. This does not lead to a
640 large reduction in runtime since all the of the high-density individuals are still need to be phased
641 to build the haplotype reference library.

642 One option to decrease runtime would be to bypass phasing completely by using the
643 high-density individuals who have been fully phased via pedigree imputation to construct the
644 haplotype reference library. We tested this in a small number of pilot simulations and found that
645 this approach reduced run time by 50%, but also decreased accuracy by 1-2%. The lower
646 accuracy is likely driven by having a less relevant set of haplotypes included in the reference
647 library, particularly from those individuals at the top of the pedigree.

648 Another option to decrease runtime of the population imputation algorithm would be to
649 decrease the number of particles that are run. We chose 40 particles for phasing the haplotype
650 reference panel, and 100 for imputing low-density individuals since those values seemed to give
651 good accuracies in pilot simulations. The number of particles used for phasing the haplotype
652 reference panel was lower than that for imputation, since errors in the haplotype reference panel
653 can be corrected in imputation, and the cost of each additional particle is higher for phasing since
654 phasing is run five times on the high-density individuals to refine the haplotype reference library.

655

656 **Approximate iterative peeling**

657 One of the goals in this paper was to improve the scaling of multi-locus iterative peeling. We
658 have previously found that multi-locus iterative peeling is a robust imputation algorithm for

659 performing imputation in large livestock pedigrees [22], but has suffered from long run-times
660 that make it impractical for regular use. We found that by approximating the full multi-locus
661 iterative peeling algorithm we were able to reduce both runtime and memory by 80% by a factor
662 while maintaining the similar accuracies. The speed improvements in AlphaImpute2, exploit the
663 fact that offspring provide relatively little information on their parent's genotype, and in many
664 cases it is possible to call the segregation values at most loci. This allows us to re-use the
665 parent's genotype probabilities in the peel-down steps for all of their offspring instead of re-
666 calculating these probabilities on an offspring-by-offspring basis, and to use lookup tables to
667 calculate the summations in the peeling-up step (particularly Equation 11).

668 In terms of accuracy the AlphaPeel and the pedigree only algorithm in AlphaImpute2 had
669 similar accuracies in both datasets. This suggests that the use of the approximations lead minimal
670 decreases in accuracy on these datasets. The primary difference between algorithms was on how
671 the founders of the population were imputed. For missing genotypes in the founders, AlphaPeel
672 imputes the individuals based on the minor allele frequency in the population. AlphaImpute2
673 imputes these individuals assuming a minor allele frequency of 0.5. We used a neutral minor
674 allele frequency in AlphaImpute2, to prevent the algorithm from incorrectly calling genotypes
675 with a low minor allele frequency for the combined algorithm, and assume that the genotypes of
676 these individuals will eventually be imputed using the population imputation algorithm. This
677 means that when run alone, the pedigree only algorithm in AlphaImpute2 may give lower
678 accuracies than AlphaPeel, but we find that the combined algorithm in AlphaImpute2 gives
679 higher accuracies than AlphaPeel in most cases.

680

681 **Comparison of population imputation algorithms**

682 Compared to the other imputation algorithms tested, AlphaImpute2 obtained generally
683 higher imputation accuracies at lower runtimes on all four simulated datasets.

684 In terms of speed, we found that AlphaImpute2 and Beagle 5.1 scaled the best out of the
685 software packages tested. findhap had initially low-runtimes on the 18k pedigree, but the
686 performance substantially decreased as the number of reference haplotypes grew larger. The
687 runtime of findhap increased from 15 minutes to 395 minutes between the 18k pedigree and the
688 107k pedigree, where the runtime of AlphaImpute2 only increased from 15 minutes to 105
689 minutes. The poorer scaling of findhap is likely due to it searching through the haplotype
690 reference library for each individual, a task that gets harder as more high-density individuals are
691 genotyped. AlphaImpute2 addresses this issue by applying the positional Burrows Wheeler
692 transform to enable constant-time searches through large haplotype reference libraries.

693 In terms of accuracy, we found that AlphaImpute2 had a higher accuracy than most of the
694 other software tested. For the pedigree only algorithm, AlphaImpute2 had a similar accuracy to
695 AlphaPeel. For the population only algorithm, AlphaImpute2 had a similar accuracy to Beagle
696 5.1 and Beagle 4.1. For the combined algorithm AlphaImpute2 had a higher accuracy than all of
697 the other algorithms including findhap and AlphaImpute. These results suggest that the
698 approximations used in multilocus peeling had limited impact on imputation accuracy for
699 pedigree based imputation, that the approximate Li and Stephens algorithm used performs as
700 well as other techniques used to fit the Li and Stephens model, and that the way that population
701 and pedigree based imputation are integrated leads to better performance compared to existing
702 software packages.

703 We were surprised by the large speed improvement between Beagle 4.1 and Beagle 5.1.
704 Beagle 4.1 had the longest runtime of any of the software packages analysed with a runtime of

705 36 hours on the 18k pedigree, whereas Beagle 5.1 had a runtime similar to AlphaImpute2 with a
706 runtime of just 28 minutes on the 18k pedigree. The improvement in speed between Beagle 4.1
707 and Beagle 5.1 are impressive, but the changes to the phasing algorithm are (to our knowledge)
708 as yet unpublished. The paper on Beagle 5 [26] only described the improvements to the haploid
709 imputation algorithm which primarily improve speed on imputing whole genome sequence data.

710

711 **Particle based approximation to the Li and Stephens model**

712 The particle based implementation of the Li and Stephens model in AlphaImpute2 takes a
713 different approach for increasing the speed of the Li and Stephens algorithm. Previous work has
714 increased the speed of diploid imputation by first pre-phasing the data, and then running a
715 haploid imputation algorithm on the phased haplotypes [18]. The split between phasing and
716 imputation is important, because for many phasing algorithms increased speed by running the
717 algorithm to directly infer the phased genotype (typically groups of heterozygous loci) instead of
718 inferring the underlying haplotypes of origin. This allowed the algorithms to scale better for
719 known loci, but means that that a haploid imputation algorithm needed to be run after pre-
720 phasing the data to fill in missing loci [35,36]. These phasing and imputation algorithms have
721 then been extended to utilize the Positional Burrows Wheeler transform to increase the speed of
722 both the phasing step [37,38] and imputation step [20].

723 In contrast, the population imputation algorithm in AlphaImpute2 runs phasing and
724 imputation together in a full diploid Li and Stephens model. In order to make this
725 computationally tractable, we approximate the Li and Stephens model using a small number of
726 particles to search for paths with high posterior probability, and use the Positional Burrows
727 Wheeler Transform to update entire sets of paths at once. Our approach is most similar to that of

728 FastLS [21], with the difference that we generate multiple (approximate) samples from the
729 posterior distribution instead of calculating a single maximum-likelihood path. This has the
730 advantage of guaranteeing a constant-time runtime for each particle, and may increase accuracy
731 if multiple paths have similar high posterior probability. We believe that techniques like
732 AlphaImpute2 and FastLS may offer an alternative avenue for performing imputation using a Li
733 and Stephens style model.

734

735 **Conclusion**

736 In this paper we present a new imputation algorithm AlphaImpute2, which combines
737 high-accuracy pedigree imputation with high-accuracy population imputation. The pedigree
738 imputation was performed by an approximate form of multi-locus iterative peeling, and the
739 population imputation was performed using a new algorithm which uses particles to approximate
740 a Li and Stephens style hidden Markov models. We find that in four simulated datasets that the
741 algorithm has higher accuracies and lower runtimes compared to other existing imputation
742 software packages, and that it scales well enough to run imputation on hundreds of thousands of
743 pedigree individuals in a matter of hours. We believe that as the size of agricultural populations
744 increase, this software will provided a much needed tool for performing imputation while scaling
745 to the datasets available.

746

747 **Declarations**

748 **Ethics approval and consent to participate**

749 Not applicable.

750 **Consent for publication**

751 Not applicable.

752 **Availability of data and material**

753 The base dataset used in this study cannot be made available due to commercial considerations.

754 However, a copy of the simulation pipeline using a purely simulated dataset is available from the
755 authors upon a request.

756 **Competing interests**

757 The authors declare no competing interests.

758 **Funding**

759 The authors acknowledge the financial support from the BBSRC ISPG to The Roslin Institute

760 BB/J004235/1, from Genus PLC, and from Grant Nos. BB/M009254/1, BB/L020726/1,

761 BB/N004736/1, BB/N004728/1, BB/L020467/1, BB/N006178/1 and Medical Research Council

762 (MRC) Grant No. MR/M000370/1.

763 **Authors' contributions**

764 AW and JMH designed the simulation study. AW carried out the simulations and analysed the
765 results. All authors contributed to writing the manuscript.

766 **Acknowledgements**

767 This work has made use of the resources provided by the Edinburgh Compute and Data Facility

768 (ECDF) (<http://www.ecdf.ed.ac.uk>).

769

770 **References**

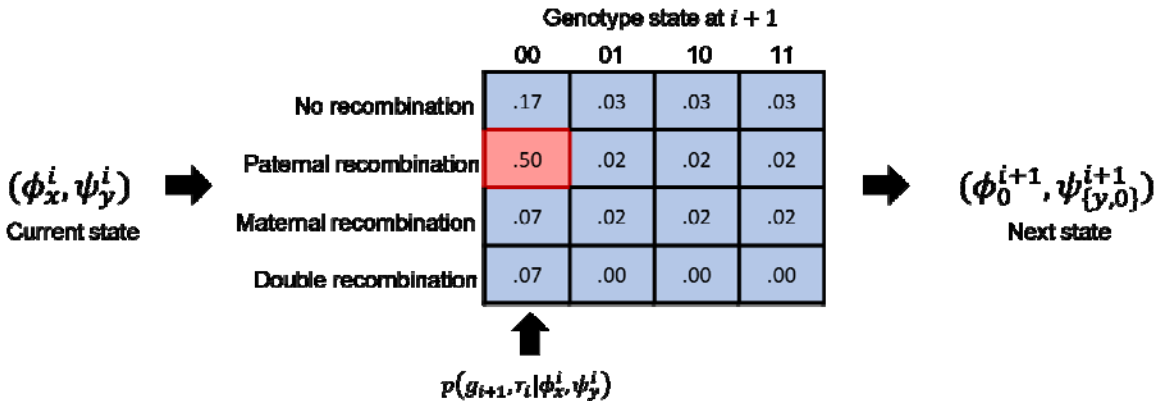
771 1. Kong A, Masson G, Frigge ML, Gylfason A, Zusmanovich P, Thorleifsson G, et al. Detection
772 of sharing by descent, long-range phasing and haplotype imputation. *Nat Genet.* 2008;40:1068–
773 75.

774 2. Li N, Stephens M. Modeling linkage disequilibrium and identifying recombination hotspots
775 using single-nucleotide polymorphism data. *Genetics.* 2003;165:2213–33.

- 776 3. Gorjanc G, Dumasy J-F, Gonen S, Gaynor RC, Antolin R, Hickey JM. Potential of Low-
777 Coverage Genotyping-by-Sequencing and Imputation for Cost-Effective Genomic Selection in
778 Biparental Segregating Populations. *Crop Sci.* 2017;57:1404–20.
- 779 4. Van Eenennaam AL, Weigel KA, Young AE, Cleveland MA, Dekkers JCM. Applied Animal
780 Genomics: Results from the Field. *Annu Rev Anim Biosci.* 2014;2:105–39.
- 781 5. Wiggans GR, Cole JB, Hubbard SM, Sonstegard TS. Genomic Selection in Dairy Cattle: The
782 USDA Experience. *Annu Rev Anim Biosci.* 2017;5:309–27.
- 783 6. Masuda Y, VanRaden PM, Misztal I, Lawlor TJ. Differing genetic trend estimates from
784 traditional and genomic evaluations of genotyped animals as evidence of preselection bias in US
785 Holsteins. *J Dairy Sci.* 2018;101:5194–206.
- 786 7. Ros-Freixedes R, Whalen A, Chen C-Y, Gorjanc G, Herring WO, Mileham AJ, et al.
787 Accuracy of whole-genome sequence imputation using hybrid peeling in large pedigreed
788 livestock populations. *Genet Sel Evol.* 2020;52:17.
- 789 8. Hickey JM, Kinghorn BP, Tier B, van der Werf JH, Cleveland MA. A phasing and imputation
790 method for pedigreed populations that results in a single-stage genomic evaluation. *Genet Sel
791 Evol.* 2012;44:11.
- 792 9. Meuwissen T, Goddard M. The Use of Family Relationships and Linkage Disequilibrium to
793 Impute Phase and Missing Genotypes in Up to Whole-Genome Sequence Density Genotypic
794 Data. *Genetics.* 2010;185:1441–9.
- 795 10. Sargolzaei M, Chesnais JP, Schenkel FS. FImpute - An efficient imputation algorithm for
796 dairy cattle populations. *J Dairy Sci.* 2011;94 (E-Suppl. 1):421.
- 797 11. VanRaden PM, O’Connell JR, Wiggans GR, Weigel KA. Genomic evaluations with many
798 more genotypes. *Genet Sel Evol.* 2011;43:10.
- 799 12. Browning SR, Browning BL. Rapid and accurate haplotype phasing and missing-data
800 inference for whole-genome association studies by use of localized haplotype clustering. *Am J
801 Hum Genet.* 2007;81:1084–97.
- 802 13. Antolín R, Nettelblad C, Gorjanc G, Money D, Hickey JM. A hybrid method for the
803 imputation of genomic data in livestock populations. *Genet Sel Evol.* 2017;49:30.
- 804 14. O’Connell J, Gurdasani D, Delaneau O, Pirastu N, Ulivi S, Cocca M, et al. A general
805 approach for haplotype phasing across the full spectrum of relatedness. *PLoS Genet.*
806 2014;10:e1004234.
- 807 15. Howie BN, Donnelly P, Marchini J. A flexible and accurate genotype imputation method for
808 the next generation of genome-wide association studies. *PLoS Genet.* 2009;5:e1000529.
- 809 16. Marchini J, Howie B. Genotype imputation for genome-wide association studies. *Nat Rev
810 Genet.* 2010;11:499–511.

- 811 17. Delaneau O, Zagury J-F, Marchini J. Improved whole-chromosome phasing for disease and
812 population genetic studies. *Nat Methods*. 2012;10:5–6.
- 813 18. Howie B, Fuchsberger C, Stephens M, Marchini J, Abecasis GR. Fast and accurate genotype
814 imputation in genome-wide association studies through pre-phasing. *Nat Genet*. 2012;44:955–9.
- 815 19. Durbin R. Efficient haplotype matching and storage using the positional Burrows–Wheeler
816 transform (PBWT). *Bioinformatics*. 2014;30:1266–72.
- 817 20. Rubinacci S, Delaneau O, Marchini J. Genotype imputation using the Positional Burrows
818 Wheeler Transform. *bioRxiv*. 2020;797944.
- 819 21. Lunter G. Haplotype matching in large cohorts using the Li and Stephens model.
820 *Bioinformatics*. 2018;35:798–806.
- 821 22. Whalen A, Ros-Freixedes R, Wilson DL, Gorjanc G, Hickey JM. Hybrid peeling for fast and
822 accurate calling, phasing, and imputation with sequence data of any coverage in pedigrees. *J*
823 *Sel Evol*. 2018;50:67.
- 824 23. Kerr RJ, Kinghorn BP. An efficient algorithm for segregation analysis in large populations. *J*
825 *Anim Breed Genet*. 1996;113:457–69.
- 826 24. VanRaden PM, Sun C, O’Connell JR. Fast imputation using medium or low-coverage
827 sequence data. *BMC Genet*. 2015;16:82.
- 828 25. Browning BL, Browning SR. Genotype Imputation with Millions of Reference Samples. *Am*
829 *J Hum Genet*. 2016;98:116–26.
- 830 26. Browning BL, Zhou Y, Browning SR. A One-Penny Imputed Genome from Next-Generation
831 Reference Panels. *Am J Hum Genet*. 2018;103:338–48.
- 832 27. Rabiner L. A tutorial on hidden Markov models and selected applications in speech
833 recognition. *Proc IEEE*. 1989;77:257–286.
- 834 28. Oliveira HR, Brito LF, Silva FF, Lourenco DAL, Jamrozik J, Schenkel FS. Genomic
835 prediction of lactation curves for milk, fat, protein, and somatic cell score in Holstein cattle. *J*
836 *Dairy Sci*. 2019;102:452–63.
- 837 29. Pocrnic I, Lourenco DAL, Chen C-Y, Herring WO, Misztal I. Crossbred evaluations using
838 single-step genomic BLUP and algorithm for proven and young with different sources of data1. *J*
839 *Anim Sci*. 2019;97:1513–22.
- 840 30. Wolc A, Drobik-Czwarno W, Jankowski T, Arango J, Settar P, Fulton JE, et al. Accuracy of
841 genomic prediction of shell quality in a White Leghorn line. *Poult Sci*. 2020;99:2833–40.
- 842 31. Chen GK, Marjoram P, Wall JD. Fast and flexible simulation of DNA sequence data.
843 *Genome Res*. 2009;19:136–42.

- 844 32. MacLeod IM, Larkin DM, Lewin HA, Hayes BJ, Goddard ME. Inferring Demography from
845 Runs of Homozygosity in Whole-Genome Sequence, with Correction for Sequence Errors. *Mol*
846 *Biol Evol.* 2013;30:2209–23.
- 847 33. Gaynor RC, Gorjanc G, Wilson DL, Money D, Hickey JM. AlphaSimR: An R Package for
848 Breeding Program Simulations [Internet]. 2020. Available from: [https://CRAN.R-](https://CRAN.R-project.org/package=AlphaSimR)
849 [project.org/package=AlphaSimR](https://CRAN.R-project.org/package=AlphaSimR)
- 850 34. Calus MPL, Bouwman AC, Hickey JM, Veerkamp RF, Mulder HA. Evaluation of measures
851 of correctness of genotype imputation in the context of genomic prediction: a review of livestock
852 applications. *Animal.* 2014;8:1743–53.
- 853 35. Delaneau O, Marchini J, Zagury J-F. A linear complexity phasing method for thousands of
854 genomes. *Nat Meth.* 2012;9:179–81.
- 855 36. Williams AL, Patterson N, Glessner J, Hakonarson H, Reich D. Phasing of Many Thousands
856 of Genotyped Samples. *Am J Hum Genet.* 2012;91:238–51.
- 857 37. Delaneau O, Zagury J-F, Robinson MR, Marchini JL, Dermitzakis ET. Accurate, scalable
858 and integrative haplotype estimation. *Nat Commun.* 2019;10:5436.
- 859 38. Loh P-R, Danecek P, Palamara PF, Fuchsberger C, A Reshef Y, K Finucane H, et al.
860 Reference-based phasing using the Haplotype Reference Consortium panel. *Nat Genet.*
861 2016;48:1443–8.
- 862



863
 864 Figure 1: A pictorial representation of a particle update. For each particle at locus i we
 865 consider a 4x4 grid of possible updates depending on the genotype state at the next locus, and the
 866 recombination state between loci. The probability of selecting of each option is given by
 867 Equation 1. In the example, the selected state, in red, is homozygous for the reference allele (00),
 868 and has a recombination on the paternal haplotype. This means that the paternal path is reset to
 869 ϕ_0^{i+1} , while the maternal path is extended to $\psi_{[y,0]}^{i+1}$.
 870

871 Table 1: Imputation accuracy, memory, and runtime for each algorithm. Beagle 4.1 did
 872 not complete within 8 days on the 63k or 107k pedigrees. Imputation accuracies were averaged
 873 across all 18 chromosomes. Time and memory are given for Chromosome 1.

	<i>High Density</i>	<i>Medium Density</i>	<i>Low Density</i>	<i>Very Low Density</i>	<i>Time (m)</i>	<i>Memory (GB)</i>
18k Pedigree						
AlphaImpute2	.998	.944	.990	.827	15	2.8
<i>Pedigree Only</i>	.998	.661	.987	.862	3	2.5
<i>Population Only</i>	.987	.929	.963	.257	13	2.1
AlphaPeel	.997	.733	.984	.855	15	9.1
Beagle 4.1	.995	.944	.969	.327	2,250	12.4
Beagle 5.1	.626	.909	.939	.219	28	17.1
AlphaImpute	.940	.875	.931	.641	348	10.5
findhap	.719	.627	.774	.445	15	4.6
34k Pedigree						
AlphaImpute2	.998	.961	.989	.699	24	4.2
<i>Pedigree Only</i>	.998	.653	.987	.843	5	4.1
<i>Population Only</i>	.992	.952	.962	.199	25	3.6
AlphaPeel	.997	.748	.986	.838	21	16.5
Beagle 4.1	.996	.959	.972	.269	4,353	17.5
Beagle 5.1	.601	.937	.936	.148	46	26.0
AlphaImpute	.955	.883	.946	.611	707	15.4
findhap	.731	.678	.787	.438	36	4.6
63k Pedigree						
AlphaImpute2	.998	.984	.991	.858	53	8.6
<i>Pedigree Only</i>	.976	.541	.974	.864	9	7.7
<i>Population Only</i>	.988	.978	.962	.188	57	6.3
AlphaPeel	.980	.596	.979	.855	74	29.8
Beagle 4.1						
Beagle 5.1	.578	.910	.937	.140	88	26.5
AlphaImpute	.898	.869	.921	.625	2,556	27.0
findhap	.757	.732	.796	.451	156	4.8
107k Pedigree						
AlphaImpute2	.998	.987	.993	.821	105	14.4
<i>Pedigree Only</i>	.991	.719	.991	.869	15	12.6
<i>Population Only</i>	.990	.981	.963	.148	106	11.6
AlphaPeel	.992	.786	.990	.858	84	50.4
Beagle 4.1						
Beagle 5.1	.682	.948	.942	.114	190	43.7
AlphaImpute	.942	.929	.940	.639	7,859	41.2
findhap	.778	.761	.801	.455	395	5.0

